Anno XI - N. 86
25 luglio 1991
Sped. Abb. Post.
Gr III/70 - CR
Distr.: Parrini

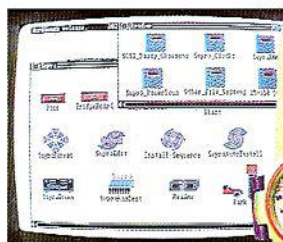
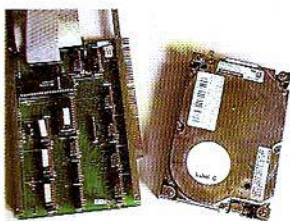
COMMODORE 64

Come rallentare un videogame

AMIGA

Tre memorie da elefante

SPECIALE

L'ora giusta per un PC

GRAFICA

- Scambio di schermate tra Amiga ed Ms-Dos
- H/Copy in assembler per Amiga

PASCAL & C

Scompattiamo i file • Quiz • Inchiesta • Enciclopedia di routine • Le migliori immagini di Bitmovie • Escape • Archivio in "C" • I migliori assemblatori per Amiga • Personalizziamo i dischetti del "C"

Leggo VR perché mi dà la rotta

Il lettore di VR è giovane, dinamico, creativo. Di cultura e reddito superiore alla media, possiede spesso più di un videoregistratore, oltre all'impianto hi-fi e al computer: nel tempo libero, non rinuncia ai viaggi in Italia e all'estero, e a cinema, teatro e spettacoli sportivi in genere. Usa il videoregistratore non solo per i programmi tv o preincisi, ma anche per riprendere i momenti felici in famiglia, per creare una videoteca personale.
E tu, che tipo di lettore sei?

VR
VIDEOREGISTRARE

IMMAGINE MARSILII

Sommario

Spazio 64

Amiga

Mondo Dos

Amiga + Ms - Dos

COMPUTER CLUB

Direttore: Alessandro de Simone

Coordinatore: Marco Miotti

Redazione / Collaboratori:
Davide Ardizzone - Claudio Baiocchi
Luigi Callegari - Umbero Colapicchioni
Donato De Luca - Carlo d'Ippolito
Valerio Ferri - Michele Maggi
Giancarlo Mariani - Domenico Pavone
Armando Sforzi - Dario Pistella
Fabio Sorgato- Valentino Spataro
Franco Rodella- Stefano Simonelli
Luca Viola

Direzione:
Via Mosè, 22 cap. 20090 OPERA (Mi)

Telefono 02 / 57.60.63.10
Fax 02 / 57.60.30.39
BBS 02 / 57.60.52.11

Pubblicità:
Leandro Nencioni (dir. vendite)
Via Mose', 22 20090 Opera (Mi)
tel. 02 / 57.60.63.10

Emilia Romagna:
Spazio E
P.zza Roosvelt, 4 cap. 40123 Bologna
Tel. 051 /23.69.79

Toscana, Marche, Umbria
Mercurio s.r.l. Via Rodari, 9
S. G.nni Valdarno (Ar)
Tel. 055 / 94.74.44

Lazio, Campania
Spazio Nuovo
Via P. Foscari, 70
cap. 00139 Roma
tel. 06 / 81.09.679

Abbonamenti: Liliana Spina
Arretrati e s/w: Lucia Dominoni

Tariffe: Prezzo per copia L. 6000
Abbonamento annuo (11 fascicoli) L. 60000
Estero: L. 100000 - Indirizzare versamenti a:
Systems Editoriale Srl
c/c 37952207 oppure inviare comune assegno bancario non trasferibile e barrato due volte a:
Systems Editoriale Srl (servizio arretrati)
Via Mose', 22
cap. 20090 OPERA (Mi)

Composizione: Systems Editoriale
La Litografica Srl Cuggiono (Mi)

Registrazioni: Tribunale di Milano
n. 370 del 2/10/82

Direttore Responsabile: Michele Di Pisa

Spedizioni in abbonamento postale gruppo III.
Pubblicita' inferiore al 70%

Distributore: Parrini - Milano

Periodici Systems:
Amiga Club (disco ed. Germania) - Banca Oggi - Computer (quotidiano) - Computer Club - 64 Club (disco) - Computer Club (disco ed. Germania) - Hospital Management - Nursing '90 - PC Club (disco ed. Germania) - Personal Computer - Jonathan - VR - Videoteca

Verso un discreto suicidio

Quando leggerete queste righe, la notizia dell'avvento delle macchine basate sui processori **80486** sarà già stata recepita e, in parte, assimilata e messa da parte.

Chi ha deciso, finalmente, di acquistare un computer Ms - Dos (e non lo ha ancora fatto), si fregherà le mani pensando che, al ritorno dalle vacanze, potrà forse procurarsi un 486 quasi al prezzo che aveva preventivato per un 386.

Chi, invece, lo ha già acquistato (e chi scrive ha speso oltre cinque milioni per un 386 acquistato nel periodo dello scorso Natale...) si starà rimproverando per la fretta avuta nella decisione.

Nel frattempo, la casa concorrente della **Intel** (la Advanced Micro Device, **AMD**) ha realizzato una specie di emulatore 386, migliore dell'originale e (sèmbra) più veloce ed economico.

Con la soluzione dei problemi relativi alla produzione del 486, in casa Intel avranno certamente tirato un sospiro di sollievo: il pericolo di un "sorpasso" da parte della concorrenza sembra, almeno per il momento, scongiurato.

Ma se pensate che, in questo momento, in casa Intel e in casa AMD si pensi ad un periodo di vacanza, vi sbagliate di grosso.

AMD, forse sorpresa (o forse no) dell'annuncio dato da Intel, ha certamente messo in moto i suoi ingegneri per la progettazione di un emulatore 486; ovviamente più economico, più veloce e contenente, magari, anche una voce digitalizzata ed un biglietto aereo per le Haway(!).

In casa Intel, invece, stanno forse cercando nuove alleanze in grado di spiazzare i successi commerciali AMD, che sono annunciati numerosi e di rilevante importanza economica.

Se, però, pensiamo che la guerra tra i produttori di chip rappresenti sempre un vantaggio, anche se indiretto, per il consumatore, ci sbagliamo. E non di poco.

Da che mondo è mondo, e gli avvenimenti dell'inizio dell'anno lo dimostrano ampiamente, a perdere la guerra è sempre stato il più debole; e non parliamo certo della *macchina bellica*, ma dei *soldati*, non tiriamo in ballo i *politici*, ma la *popolazione*.

E noi utenti, nella similitudine, a quale delle due "parti" somigliamo?

Lo stesso abbassamento dei prezzi, infatti, può rappresentare un'arma a doppio taglio che se, da una parte, ci consentirà di acquistare prodotti a prezzi inferiori, costringerà, dall'altra, gli importatori ad effettuare manovre economiche o tecniche (o entrambe) tendenti a limitare la qualità dei servizi e, in definitiva, a danneggiare l'utente.

Che cosa possa succedere, in un mercato già in crisi, è difficile da prevedere, tanto che rimane un'incognita perfino per gli esperti del settore.

Ed è proprio per questo che dovremmo preoccuparci di più...

Alessandro de Simone

Trasparenza

Che cosa si intende, esattamente, con il termine "trasparenza"?
(da alcune lettere)

Quando si impartisce un comando, un *qualunque* comando, il computer (o meglio, il programma in quel momento attivo) esamina ciò che abbiamo digitato ed esegue, se possibile, i compiti dovuti.

Impartendo, ad esempio, il comando **Dir**, il computer attiva un programma in linguaggio macchina che, tra l'altro, prende in considerazione vari fattori: effettiva presenza del disco nel drive specificato, esame della FAT, attivazione delle routine in grado di visualizzare i nomi dei file e delle directory, eventuale scroll dello schermo quando la visualizzazione giunge alla fine del video, esame continuo della tastiera per verificare se l'utente ha premuto i tasti **Ctrl** + **C**, eccetera.

Tutte queste funzioni (e tante altre, sulle quali non ci soffermiamo) vengono svolte dal computer senza che l'utente se ne accorga, in modo quindi **"trasparente"**, nel senso che si svolgono senza dare alcun segno della loro presenza.

Il termine informatico, pertanto, ha un significato diametralmente opposto a quanto comunemente si intende; ad esempio, in politica, per operazione trasparente si intende un'operazione i cui dettagli possano esser noti a tutti, nei minimi dettagli.

Mi viene un sospetto: vuoi vedere che, nel caso ad esempio di **Gladio**, la "trasparenza" di cui si parlava era intesa in senso informatico?...

Sillabazione

Al programma di sillabazione, apparso nella sfida del mese del N. 83, ritengo che possa essere apportata una modifica che impedisce la visualizzazione del trattino anche dopo l'ultima sillaba.

E' sufficiente, allo scopo, aggiungere le righe 35, 265, 275 e modificare le righe 30, 260 e 270:

```
30 Print: Print: Input
"Parola (0=fine)"; A$:
S1 = Len (A$)
35 If S > S1 Then Goto
20
260 Print Mid$(A$, S,
1);: If S+1 < S1 Then
Print "-";: S=S+1:
Goto 35

265 Goto 20
270 Print Mid$(A$, S,
2);: If S+1 < S1 Then
Print "-";: S=S+2:
Goto 35
275 Goto 20
```

Il resto del programma, ovviamente, deve esser lasciato inalterato.
(A. Zamparelli - Benevento)

LA VOSTRA POSTA

(a cura di A. de Simone)

Modem e Fax

E' iniziata la distribuzione di nuovi modem ad alta integrazione. Distribuiti dalla Evart di Milano (tel. **02/48.14.61.9**) offrono caratteristiche di un certo rilievo a prezzi competitivi.

Ci riferiamo, in particolare, ai modelli **9624P** (Pocket) e **9624H** (Half size card) che racchiudono numerose funzioni: Fax ricetrasmittenti (9600 bps), modem da 2400 baud, basso consumo, numerosi standard (V21, V22, V22 bis, V27 ter, V29, VT.30/t4); le confezioni contengono programmi di telecomunicazione sia Fax che modem.

Alcuni modelli offrono la possibilità di correzione d'errore, aumentando in tal modo l'affidabilità dei circuiti.

La serie di modem, interni ed esterni, è offerta presso numerosi rivenditori specializzati, il cui elenco è disponibile telefonando alla sede di Via Rossetti 17 (Milano)

Lui non sa chi sono io

Da anni seguo con ammirazione la vostra pubblicazione e, sempre grazie a voi, non solo sono passato dal C/64 all'Amiga e all'Ms-Dos (con enorme soddisfazione), ma ho deciso di iscrivermi alla facoltà di informatica. Ho solo una richiesta: potete pubblicare la foto dello staff di Computer Club e, in particolare, del direttore Alessandro de Simone?
(Emanuele R. - Mar.ta di Savoia - FG)

Sono contento per la "fedeltà" alla testata; per ciò che riguarda l'iconografia, ho deciso di svelare finalmente un segreto: i nomi degli autori che corredano gli articoli della rivista sono solo pseudonimi, dietro i quali si nascondono graziose fanciulle.

Siamo già comparse spesso e volentieri sulle pagine della rivista, anzi, poiché alcune nostre foto gettano lo scompiglio tra vari lettori (vedi Ernesto il censore, Posta del N. 85) saremo costrette a pubblicare, al posto delle nostre vere immagini, fotografie di personaggi insospettabili, tra cui un ingegnere, un ragioniere ed un geometra.

Troppo tardi

Vi invio alcuni programmi in Im, realizzati in tre anni di lavoro, che aggiungono 74 nuovi comandi al Basic del C/64, definiscono fino a 120 sprite (anche in multicolor) e consentono di caricare schermate grafiche del computer Sinclair Spectrum.
(Giacomo F. - Maresca)

Purtroppo la pubblicazione di dette routine occuperebbe l'intera rivista e non sarebbe, per giunta, molto utile ai nostri lettori a causa della disponibilità, nel circuito degli appassionati, di decine di utility del genere descritto. E' un vero peccato, tuttavia, che una tale bravura nel campo della programmazione Im (solo chi è veramente bravo, infatti, può essere in grado di portare a termine un lavoro del genere) sia limitata al C/64.

Non so se il "passaggio" ad un sistema superiore sia impedito da ristrettezze economiche; in caso contrario ti consiglio di cambiar computer per intraprendere, finalmente, un'attività realmente professionale, che un programmatore del tuo livello sicuramente merita.

Travi e pilastri

Vi invio, come da accordi telefonici, un gruppo di programmi in Gw - Basic per la determinazione di travi e pilastri in cemento armato.
(Giuseppe Cannella)

Purtroppo i listati inviati non sono corredati da alcun articolo esplicativo nè è presente un minimo esempio di applicazione degli stessi. Ciò che era stato richiesto non era una semplice raccolta di routine di ingegneria, ma qualcosa di molto più impegnativo, tale da interessare i lettori appartenenti al ramo specifico (geometri e ingegneri). Ritengo, tuttavia, che il lavoro svolto sia interessante e merita la pubblicazione, a patto di completarlo come concordato.

Una telefonata, a questo punto, è d'obbligo.

Proteste di un 64ista

Sono un affezionato lettore perché gli argomenti che trattate mi piacciono e mi piace il modo in cui ve ne occupate. Tuttavia non sopporto alcune vostre risposte nelle pagine dedicate alla posta, di dubbio gusto, che da un po' di tempo date ai lettori che chiedono consigli.

Ammettiamo anche che, a volte, le domande sono forse un po' ingenue, ma ritengo che sarebbe meglio non rispondere affatto piuttosto che essere scortesi o fare dell'ironia fuori luogo.

Mi riferisco alla risposta apparsa sul n. 83, a proposito di un lettore che chiedeva consigli sulla cartuccia da acquistare per il suo C/128.

Forse voi ignorate che qualcuno possa aver fatto dei sacrifici per comprare un certo computer, con rispettive periferiche, ed anche se (perché inesperto o influenzato dal negoziante) ha scelto un modello superato o poco valido, è pur sempre frutto dei suoi risparmi.

Già è frustrante il fatto di avere un C/128 e di usarlo pressoché unicamente in modo 64 poiché la disponibilità di programmi per questo computer è quella che è. Se, poi, chiedendovi consiglio, vi mettete a fare facile ironia (senza peraltro rispondere alla domanda) dicendo che è un sistema da rinchiudere perché obsoleto, la cosa diventa veramente avvilente.

Perciò, prima di rispondere, pensate che chi vi scrive è un lettore (e quindi **compra** la vostra rivista) e se vi pone una domanda, per quanto banale essa sia (ammesso che lo sia) è perché si fida di voi.

Non ultimo motivo di questa mia lettera è il far presente che non tutti si possono permettere, economicamente parlando, un sistema superiore a quello posseduto, considerando anche il fatto che rivendere un computer un po' obsoleto non è facile. Per questi motivi penso che dovreste essere più attenti nelle risposte e siccome nessuno è un "Dio" del sapere, se qualche volta non sapete che cosa rispondere, dite semplicemente "non lo so".

Concludo dicendo che tali critiche non possano che rendere migliore la vostra (nostra) rivista, altrimenti ritenetele solo come punto di vista di un vostro lettore. Cordialmente vi saluto, sperando, e soprattutto credendo, in una vostra risposta che, sono sicuro, non mancherà

Andrea Chechi - Grosseto

Lungi dal fare ironia o, peggio, dal prendere in giro, lo scopo della risposta in questione era unicamente quello di invitare il lettore a riflettere (e molto) prima di continuare ad investire in un computer ad 8 bit.

*Un vero computer **Ms - Dos** o un **Amiga**, infatti, oggi costano quasi quanto tre o quattro cartucce per **C/64** o **C/128**; in occasione del prossimo **Smau** prevediamo un ulteriore calo dei prezzi oppure, (il che è lo stesso) un mantenimento degli attuali prezzi ma la contemporanea offerta di modelli tecnologicamente più evoluti.*

Per quanto riguarda i sacrifici economici, poi, non mi stancherò di ripetere ciò che dico molto spesso da un po' di tempo a questa parte: tutto ciò che si poteva dire sul "sistema" C/64 (il "sistema" C/128, purtroppo, non è mai esistito) è stato già detto: scrivere ancora sull'argomento significherebbe ripeterci e, in pratica, prendere in giro il lettore 64ista.

*Del resto nella stessa lettera, in più punti, si parla di **obsolescenza** e di implicito riconoscimento dell'inadeguatezza di un sistema ad 8 bit nell'attuale momento "storico" che sta vivendo l'informatica.*

di Alessandro de Simone

Manuale di confusione

Non sempre i manuali sono chiari. Cerchiamo di capire dove e, soprattutto, perché manca la dovuta chiarezza

Chi decide di acquistare un programma professionale spesso non ha altre informazioni se non quelle, accattivanti, che traspaiono dal cellofan della voluminosa confezione presente sugli scaffali dei rivenditori.

I messaggi contenuti sui corposi parallelepipedi cellofanati sono di notevole interesse sia per quanto riguarda la qualità del prodotto...

"...il più potente package professionale mai realizzato... la miglior creatura software del nostro secolo... dopo di lui c'è solo Lui..."

...sia per ciò riguarda le potenzialità messe a nostra esclusiva disposizione dall'altruistica software house...

"...consente l'importazione e l'esportazione da/verso altri formati grafici... permette di incrementare oltre misura la Vostra produttività... basta con l'inefficienza e la lentezza..."

L'osservatore più attento, cioè quello che non si limita solo a spalancare la bocca nel leggere le caratteristiche del software, noterà certamente che la vetrina di qualsiasi negoziante di computer contiene tanto e tale software che si potrebbero cambiare i destini del mondo da così a così.

Purtroppo, tornati a casa con il pesante fardello di dischetti e manuali, ci si accorge ben presto che l'evoluzione delle abitudini del pianeta, e soprattutto la nostra, subirà qualche ritardo rispetto alle rosee aspettative.

Tra il leggere e l'imparare

Non mi è mai capitato di conoscere personalmente l'autore di qualche manuale; spesso ho cercato di immaginare le caratteristiche mentali di un traduttore standard, e non sempre in termini benevoli.

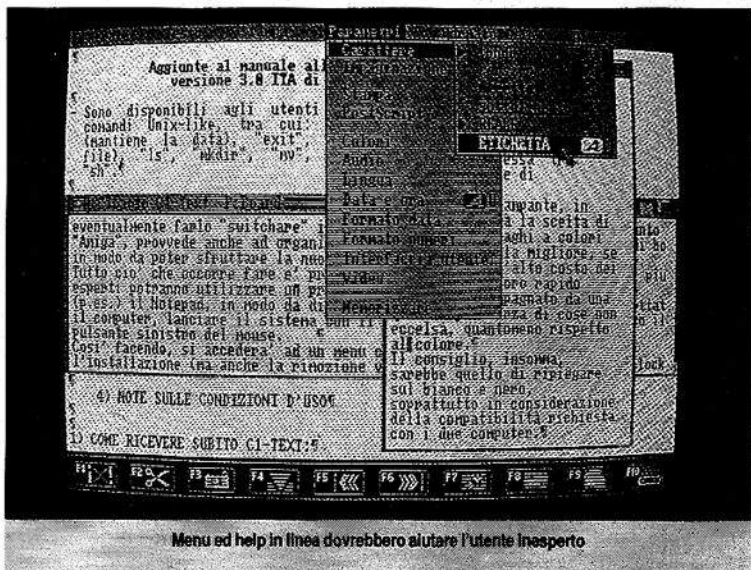
Ricordo che, agli albori dell'informatica, tutta la manualistica era reperibile solo in lingua inglese. Una volta mi capitò di esaminare i due volumi (l'originale, in inglese, e la corrispondente traduzione in italiano), acclusi nella confezione di un programma: erano i primi, timidi tentativi di andare incontro agli utenti italiani.

Felice della novità, trascurai il manuale in inglese e mi dedicai interamente all'altro.

Questo, più o meno, era un misero tentativo di contrabbandare, come lingua italiana, un'accozzaglia indistinta di termini sgrammaticati, legati tra loro grazie ad una sintassi molto personalizzata e ad una logica disinvolta.

La modestia che contraddistingue il mio carattere, tuttavia, addebitava la mancata comprensione alla mia inesperienza.

Mi arresi solo quando, nel commento di un listato Assembly, incontrai una frase che suonava più o meno così:

Menu ed help in linea dovrebbero aiutare l'utente inesperto

"...il riconoscimento della bandierina viene effettuato in maniera automatica..."

Che diavolo era 'sta bandierina? Incuriosito cercai, nel manuale originale in inglese (provvidenzialmente incluso nella confezione) il passaggio corrispondente: immaginare la mia sorpresa nel constatare che la **"bandierina"** altro non era se non un **Flag**!

Buttato il "manuale" italiano nel cestino della carta straccia, la vita tornò a sorridermi.

Da ieri ad oggi

Oggi, per fortuna, non solo il Flag rimane Flag, ma tutti gli altri termini tecnici vengono lasciati integri nella propria dignità. Se qualche appunto deve esser messo in evidenza, invece, questo va mosso nei confronti dell'atteggiamento mentale di tre persone: l'autore del manuale, il traduttore, l'editore. Quest'ultimo, innocente solo in apparenza, ha infatti la grande responsabilità di selezionare il traduttore cui affidare il lavoro di conversione.

L'autore del programma

Chi scrive programmi, purtroppo, è un maniaco inevitabilmente staccato dal mondo e, soprattutto, dalla realtà che lo circonda.

Chi è in grado di scrivere da 2 a 3 mega di byte di programma, infatti, (anche se i file sono, in effetti, il risultato di un lavoro di compilazione) non può essere normale, nè può frequentare gente normale.

Si incontra con programmatori, segue fiere del settore, è in stretto contatto con manager che usano il computer. Tali abitudini, a lungo andare, gli fanno ritenere che l'utente medio (o meglio, il potenziale acquirente dei suoi programmi, o meglio ancora, il cittadino umano medio) è un maniaco come lui e che, come tale, sa tutto di tutto.

E' un po' come pretendere, dagli studenti, di essere già preparati su vari argomenti, trascurando, invece, di considerare che uno studente va a scuola proprio perché (senza offesa per nessuno), è da considerare *tecnicamente* ignorante (altrimenti, perché andrebbe a scuola?).

In certi casi, per "fortuna", allo staff tecnico della software house in cui lavora il geniale programmatore, appartiene qualcuno che si occupa degli altri (forse era un boy scout, ha studiato sociologia oppure, più semplicemente, è un neo-assunto). Questo personaggio, talvolta, riesce a convincere l'autore sull'opportunità di introdurre qualche chiarimento su vari passaggi oscuri che un potenziale utente potrebbe incontrare nella lettura del manuale.

I programmi di grafica contengono termini tecnici inusuali

Forse è questa la causa della stramberia propria di alcuni paragrafi, come questo che segue:

"Premendo il tasto Control (di colore solitamente grigio, presente in basso a sinistra sulla tastiera del computer che state usando in questo momento) ed il tasto "S" (iniziale di Savona, per l'utente italiano) è possibile registrare su dischetto la lettera che avete digitato. Per inserire un file IFF è però necessario dapprima escluderne la parte eventualmente grafica mediante l'apposita utility da attivare nella subdir \c\define\arc avendo avuto l'accortezza di portare a 22 il numero di file presenti in Config.sys".

Aiuto! Che cavolo significa? Perché mai dilungarsi sul colore del tasto Control (inserendo addirittura una foto della tastiera, con tanto di freccia che lo indica) se poi, subito dopo, si parla in dialetto ostrogoto?

E ancora, avete mai visto un Help in linea, tanto di moda da un po' di tempo? Esempio: nel menu di un (costoso) s/w è possibile selezionare, tra le altre, una "voce" che, a causa del limitato spazio a disposizione, si limita a dire *"Annulla com."*; premendo il tasto di Help compare l'esauriente spiegazione: *"Annulla il comando"*. Ogni commento, sull'utilità di una simile spiegazione, è superfluo. Sarebbe auspicabile, e più onesta, la visualizzazione di un altrettanto breve, ma più significativo, messaggio del tipo: *"Vedi manuale a pagina XXX"*.

Con questo non vogliamo affatto dire che il programmatore è cattivo o se ne frega del suo cliente, ma semplicemente affermare che l'autore del manuale deve esser diverso dall'autore del programma. Quest'ultimo dovrebbe limitarsi a consegnare alcuni appunti generici, oppure avere solo incontri verbali con chi è incaricato di scrivere il manuale.

Quest'ultimo, inoltre, dovrebbe essere un neofita nel campo dell'informatica: solo in questo modo, incontrando punti oscuri (che verrebbero inevitabilmente giudicati tali anche dal futuro acquirente del programma) si vedrebbe costretto a chiedere spiegazioni, e a riportarle nel manuale.

Per sapere esattamente come scrivere un manuale si dovrebbe pedinare, di nascosto (per garantire la spontaneità), un cliente che, dopo aver acquistato un programma, si reca a casa sua, magari in

I PICCOLI COMMODORE SONO CRESCIUTI

compagnia di un amico, per metterlo in funzione (il programma, non l'amico).

Sistemata una telecamera ed un microfono nei pressi del computer, si potrebbero osservare le reazioni dei due utenti, i loro commenti e, magari, cronometrare il tempo di apprendimento e di soddisfazione delle varie fasi (installazione, primi rudimenti, applicazioni consuete, applicazioni avanzate).

Tutto ciò, come già detto, dovrebbe esser realizzato di nascosto e preso in seria considerazione per la stesura dei futuri manuali.

Una mal interpretata libertà dell'individuo (pare che sia vietato pedinare gente e piazzare microfoni e telecamere nelle abitazioni altrui) costringe gli autori ad "immaginare" soltanto le reazioni dell'utente ed a scrivere i manuali limitandosi a sperare che siano comprensibili.

L'autore del manuale

In effetti, parlando dell'autore del programma, siamo stati costretti a descrivere anche le caratteristiche dell'autore del manuale (talvolta, purtroppo, sono la stessa persona).

Per una esagerata generalizzazione del detto *"repetita iuvant"* (e, soprattutto, sfruttando in modo indegno le funzioni di *"taglia"* e *"incolla"* offerta da tutti i word processor) oltre il 20% delle pagine di un manuale viene riempito da frasi ripetute quasi all'infinito.

Avvertenze circa la...

"...possibilità, a questo punto delle operazioni, di interrompere l'azione in corso premendo il tasto Esc: compare un request (finestra) in cui è necessario cliccare su Yes per confermare, su No per annullare la richiesta erroneamente impartita..."

...compaiono in corrispondenza di **tutti** i comandi rintracciabili in **tutti** i menu a discesa; e non sono pochi. Basterebbe un intero capitolo, all'inizio, che descriva, tra le altre, la funzione del tasto Esc e farvi riferimento nel testo costantemente (esempio: *"per annullare il comando vedi Cap. 1 Par. 3"*).

Alla stessa famigerata categoria degli autori possono, a buon diritto, appartenere anche i traduttori.

Questi elementi, che spesso svolgono un'attività legata esclusivamente al mondo letterario (in cui vivono individui abituati a guardare con disprezzo i tecnici, prima di voltarsi per vomitare), traducono in modo letterale tutti i termini che non riescono a comprendere.

Ricordo che una volta dovetti litigare ferocemente con uno di questi elementi che, incaricato di tradurre un manualetto di programmazione in Basic per principianti, aveva modificato tutti gli apici ("), che delimitavano le stringhe Basic degli esempi riportati, nei caratteri minore-minore («) e maggiore-maggiore (»), perché, secondo lui, gli apici *"sono brutti e non si usano"*.

E pensare che, all'inizio del manualetto originale, era messo in risalto (ed era stato quindi tradotto dalla stessa persona) il particolare che, in Basic, le stringhe vanno *sempre* racchiuse tra apici!

Non ho più incontrato quella persona, nè penso che qualcuno l'abbia reclamata a *"Chi l'ha visto?"*...

Un linguaggio presenta spesso difficoltà di comprensione

L'editore del manuale

L'editore, poveretto, non può fare altro che fidarsi dell'esperienza altrui.

Si rivolge, quindi, a persone che siano in grado, più o meno, di parlare inglese o che vantano tale caratteristica.

Eppure vi sono moltissimi interpreti e traduttori che conoscono l'inglese tecnico (ed informatico in particolare) in modo pregevole.

In particolare, alle conferenze stampa della **MicroSoft** (i cui oratori parlano prevalentemente in lingua inglese) vengono incaricate della traduzione simultanea due traduttrici che non solo traducono in modo chiaro e perfetto (e, badate bene, in tempo reale), ma si permettono addirittura il lusso di spiegare alcune frasi idiomatiche e giri di parole, riuscendo sempre a "riprendere" gli oratori che, forse ignari della manipolazione delle loro parole, continuano a parlare a gran velocità.

Nel mercato attuale, definito "in crisi" da più di una persona competente, l'unico modo di tornare agli antichi fasti è quello di offrire serietà e qualità di servizi, ovunque sia possibile.

E non ditemi che un manuale non può esser considerato un "servizio" da usare come arma letale per limitare i danni della concorrenza e della pirateria...

In caso di guasto... cambiare il computer!

Che cosa ci può capitare se il computer richiede una riparazione

Una lettera giunta in Redazione ci ha lasciato piuttosto perplessi:

"Il titolare di un importante negozio di computer della mia cittadina (peraltro mio intimo amico) ha affermato che la tavoletta grafica che stavo per acquistare poteva vendermela senza garanzia e mi metteva in guardia da un simile acquisto. E' possibile una cosa del genere?"

Da un punto di vista legale, teniamo a precisare, l'affermazione è del tutto falsa dal momento che qualunque prodotto venduto in Italia, se in mancanza di altre specifiche, è garantito, per legge, per un anno a partire dalla data del suo acquisto. Con alcuni trucchetti, tuttavia, è possibile aggirare la legge. Anzitutto, grazie alla tradizionale lentezza burocratica (che consente la scarcerazione di folle di mafiosi in tempi, quelli sì, ristrettissimi) nessuno mette in mezzo un avvocato per far rispettare diritti vantati su un apparecchio il cui valore è minore della parcella del professionista.

In secondo luogo è possibile ridurre artificiosamente la durata della garanzia, ad esempio trattenendo per numerosi mesi l'apparecchio inviato in riparazione.

Dal momento che non si può dimostrare, inoltre, che l'apparecchio non è stato manomesso, alcuni centri di riparazione possono approfittarne per annullare o limitare la garanzia, almeno nei casi dubbi.

La garanzia si può ulteriormente ridurre prendendo in considerazione non la data di acquisto dell'utente finale, ma quella in cui è stata ceduta al distributore o al negoziante.

Naturalmente a simili espedienti ricorrono le ditte meno serie che pur di far soldi sarebbero disposti a vendere la propria madre (senza, ovviamente, garanzia).

Del resto, le responsabilità in gioco non sono mica da ridere. In molti casi non è semplice dimostrare che l'apparecchio si è rotto per "vizi" di origine e non per incuria dell'utente.

Certo è che i centri di assistenza raramente preferiscono lavorar gratis.

Un joystick usato... intensamente; quale laboratorio potrà mai ripararlo?

Tecnologia

Nel campo dell'informatica i progressi compiuti sono tanti e tali che, da una parte, l'integrazione e la qualità raggiunte consentono una miniaturizzazione praticamente folle; da un'altra parte, invece, grazie alla disponibilità di moderni strumenti di laboratorio, è molto più semplice e rapido di una volta individuare quale dei, pochissimi, chip presenti su una scheda è guasto.

Il problema è sostituire il componente difettoso dal momento che molto raramente si tratta di sostituire componenti passivi (resistenze, condensatori) di semplice rimozione.

E' infatti necessario avere a disposizione speciali dissaldatori per rimuovere i chip (sempre saldati sulla piastra, e quasi mai su zoccolo, per questioni di affidabilità di funzionamento), operazione che si presenta sempre rischiosa per le "piste"

di rame dei circuiti stampati che, in caso di rottura, sono da buttar via.

L'operazione di riparazione, quindi, può venire a costare molto di più della scheda nuova.

Per fortuna i costi di produzione, costantemente in ribasso, consentono di effettuare la sostituzione dell'intera scheda a prezzi oltremodo contenuti.

Una moderna Super VGA; quanto può costare ripararla?

Cambio scheda

Conosco un commerciante che ha risolto il problema della garanzia, e dell'assistenza, in modo molto radicale, oltre che semplice ed economico (per lui ma anche, in definitiva, per l'utente).

Se gli portano, ad esempio, un computer che non riesce più a registrare o a leggere dal drive, egli non fa altro che aprire il contenitore e sostituire prima il controller, poi il drive (o tutti e due) con alcuni esemplari che normalmente vende ai propri clienti.

Di solito, appunto, c'è qualcosa che non va nel drive o nel controller (o in tutti e due).

Se, appunto, è il drive che non funziona bene, il negoziante in questione fa osservare al cliente l'estrema miniaturizzazione della scheda e dice più o meno così:

"Se vuole inviare il drive per la riparazione, questa non le costerà meno di 200 mila lire (individuazione chip guasto, rimozione, acquisto pezzo di ricambio, inserimento, verifica, eccetera) senza contare il tempo necessario alla riparazione stessa, durante il quale sarà costretto a fare a meno del computer; se, invece, acquista un drive nuovo, questo le costa solo 180 mila lire."

Naturalmente il discorso viene fatto soltanto a clienti nuovi (ci mancherebbe altro!) oppure nel caso di prodotti la cui garanzia è già scaduta; nel caso, invece, in cui il guasto si è verificato su un computer acquistato presso il negozio del mio amico, la sostituzione dell'apparecchio "guasto" con il "sano" viene effettuata sul posto, senz'altra spesa o fastidio, se non quello di aver portato il computer in negozio.

Una scheda seriale, anche se di basso prezzo, si rompe molto raramente!

"Vedi -mi ha confidato un volta il mio amico- *prima pretendevo dal distributore la garanzia assoluta sui pezzi acquistati, ed il prezzo era ovviamente proporzionale; quando mi sono accorto che i computer e gli accessori si guastavano con una frequenza media inferiore all'1%, ho chiesto, al distributore, di praticare uno sconto sui prodotti che acquistavo; in cambio rinunciavo totalmente alla garanzia. In pratica ho ottenuto uno sconto del 7% che, considerando che le sostituzioni (ovviamente a mie spese) si mantengono mediamente sulla cifra dell'1%, rappresenta una diminuzione reale dei costi del 6%, che mi consente di vendere prodotti a prezzi inferiori a quelli praticati dalla concorrenza, che per le riparazioni si affida invece al servizio offerto dal distributore".*

Oggi, infatti, una scheda XT costa meno di 100 mila lire; una intera scheda base 386 dovrebbe costare, entro la fine di quest'anno, meno di 500 mila lire (prezzi, ovviamente, per quantità rilevanti).

Non sarà più conveniente, tra breve, effettuare riparazioni, ma sarà molto più vantaggioso buttare via il computer vecchio e, con i soldi preventivati per la riparazione, acquistarne uno nuovo, più moderno.

L'anno scorso, ricordo a tal proposito, fui costretto a portare presso un centro di assistenza il mio primo computer AT che, acquistato nel 1987, era costato un bel

po' di quattrini (era uno dei primi 80286 commercializzati).

Purtroppo il componente che si era guastato (dal prezzo inferiore alle 10 mila lire, almeno a quanto risultava dall'ultimo listino rintracciabile) non era più prodotto e si rendeva necessaria la sostituzione dell'intera scheda (non standard, data la vetustà del computer) con una di nuova produzione, ma anch'essa non standard. Morale della favola: quasi un milione di lire, IVA esclusa, per la semplice rottura di un maledetto, obsoleto chip!

A quei tempi, purtroppo, un intero computer AT costava ben più di un milione, e preferii farlo riparare. Oggi, ovviamente, se dovesse rompersi ancora, lo butterei alle ortiche.

Un chip su zoccolo consente una economica sostituzione

La madre di tutte le schede

Il discorso della sostituzione più conveniente della riparazione può esser valido soltanto se, appunto, il computer di cui parliamo è formato da schede, tutte standard o, comunque, di elevata diffusione.

I computer Ms - Dos compatibili di elevata diffusione sono costituiti da una **scheda madre** in cui sono alloggiati solo i chip necessari alla gestione della "macchina" 80x86. Vale a dire il microprocessore, l'eventuale zoccolo del coprocessore matematico, gli zoccoli per le Ram, i connettori per le schede aggiuntive.

In pratica sono assenti i chip della scheda video, delle porte parallela e seriale, del controller.

In questo caso, se si guasta la scheda madre, sarà possibile sostituirla con una simile (anche di altra marca, magari più economica...) avendo l'accortezza di rimuovere le memorie (se ancora valide), il coprocessore matematico (ma solo se presente su zoccolo), il controller: lo standard oggi adottato da tutti i fabbricanti, infatti, permette la connessione con schede di ogni marca.

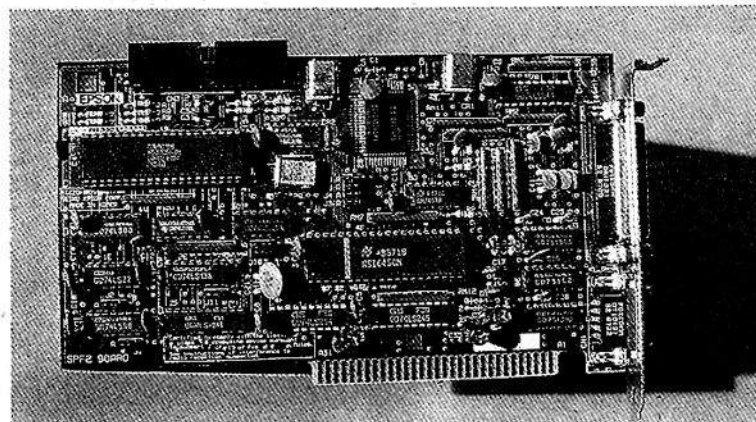
Se, però, la scheda principale contiene altre circuiterie (porte varie, gestione video) la sostituzione è ancora ovviamente possibile ma, inevitabilmente, l'operazione costerà di più a causa della presenza degli altri chip.

Anche nel caso di guasto della sola scheda video, ad esempio, la riparazione-sostituzione di una VGA costituirà meno problemi (soprattutto economici) se questa è "isolata" o se, invece, fa parte della scheda madre; senza tener conto del fatto che, in caso di guasto, si può decidere (dato che ci siamo) a sostituirla con una più moderna, magari con una super VGA invece che con una normale VGA.

Tale miglioramento, ovviamente, non potrebbe esser possibile se la circuiteria video, presente sulla scheda madre sostituita, è sempre quella originale.

Anche in caso di guasto del controller potrebbe esser vantaggioso sostituirlo con uno diverso, che magari consente il collegamento ad un numero maggiore di drive; e così via. In alcuni casi, tuttavia, la presenza di tutti i chip di un computer sulla scheda madre è indispensabile.

Ci riferiamo, come intuitivo, ai **computer portatili** che, per forza di cose, non possono permettersi il lusso di montare connettori, e relative schede, che occuperebbero spazio (e peso) preziosissimi.

Ma per un computer da tavolo l'esigenza della miniaturizzazione potrebbe non avere molti sostenitori. Ecco quindi, che prima di acquistare un computer sarebbe opportuno, dopo aver dato uno sguardo ai prezzi, effettuare la scelta su un modello che, in caso di guasti (sempre toccando ferro, per carità!) possa darci il minor numero di problemi possibile.

La riparazione di un vecchio controller può risultare non conveniente rispetto alla sua sostituzione

di Massimiliano Alessandri

Come ti muovi ti cancello

Una procedura, in ambiente Amiga, consente di spostare intere directory (e loro contenuto) da un dischetto ad un altro

Se avete necessità di trasferire una subdirectory (copiandola da un disco ad un altro e poi cancellandola dall'originale) e, possedendo un solo drive, trovate quest'operazione poco pratica, continuate a leggere queste pagine, che rappresentano la risposta alla "sfida del mese" lanciata nel numero **84**.

La sfida, lo ricordiamo, consisteva nel realizzare un **batch file** in grado di eseguire tale operazione effettuando, nel contempo, una rigorosa verifica sulla sua reale fattibilità, astenendosi dal cancellare la directory originale in caso di copia malriuscita della stessa.

Il programma pubblicato implementa su Amiga la funzione MUOVI che, per la sua semplicità strutturale, si presta ad un agevole studio anche da parte di chi si è avvicinato da poco ad Amiga e, magari, possiede solo 512 K di memoria Ram ed il solo drive interno.

Per utilizzare Muovi sono necessari i soli comandi contenuti nella directory **C** del disco **Workbench** poiché esso **non** fa uso di utility di pubblico dominio di vario tipo, come **DirCopy**, **DirErase** o simili.

E' preferibile memorizzare il batch nella directory **S**, redigendolo con un qualunque editor, purché in grado di salvare file in formato ASCII, e successivamente impartire un comando del tipo...

```
PROTECT S:Muovi +S
```

...per renderlo autoeseguibile da Shell senza ricorrere ad EXECUTE.

La sintassi "completa" da usare, comunque, è la seguente:

```
EXECUTE MUOVI PathSorgente
NomeDirectory PathDestinazione
```

...(da scrivere su un solo rigo, ovviamente) in cui **PathSorgente** e **PathDestinazione** possono essere nomi di dischi (ad esempio **Workbench1.3:**), devices (es. **DF1:**, **DH0:**) o percorsi logici (es. **"MyDisk: Programs/"**, notare le virgolette).

Se i tre nomi contengono **spazi** (come, appunto, nell'ultimo esempio), sarà necessario, come è noto, racchiuderli tra doppi apici (") secondo le regole di AmigaDOS.

Come gira

Ma veniamo alla spiegazione del funzionamento del programma.

La prima pseudo-istruzione **.Key** preleva, dalla riga di comando, i **tre** parametri menzionati, memorizzandoli in altrettante variabili.

Poco da dire sulla successiva (banale) sequenza di istruzioni COPY, che memorizza in RAM: i sette comandi necessari al batch, allo scopo di velocizzare le operazioni (e diminuire drasticamente il numero di scambi di dischetto necessari).

Innalzando la soglia d'errore con l'istruzione FAILAT evitiamo che gli eventuali errori occorsi durante le operazioni blocchino il programma, permettendoci di gestirli nel flusso di programma.

Successivamente la directory viene copiata dal disco sorgente in un buffer nel disco RAM. Se l'operazione non riesce, viene emessa una segnalazione d'errore. Questa potrebbe essere causata da un'insufficiente disponibilità di memoria (il disco RAM non è in grado di contenere la directory) da un errore sul disco sorgente o dall'inesistenza della directory stessa. Se tutto va liscio, e se non esiste sul disco di destinazione una directory con lo stesso nome, si passa alla copia effettiva.

Un messaggio d'errore potrebbe essere visualizzato se il disco sul quale si tenta di trasferire la directory è pieno; contiene errori di vario tipo o non è formattato; è protetto contro la scrittura; non esiste.

In questo caso l'operazione di copia si interrompe e si provvede a cancellare ciò che è stato eventualmente copiato.

In seguito, SOLTANTO se il trasferimento è stato realmente effettuato senza errori, la directory sul disco sorgente viene completamente cancellata.

Alla fine della procedura il disco RAM viene completamente liberato (ciò avviene anche in caso di errori nella procedura di trasferimento).

Una nuova sfida

Il lavoro svolto dal nostro lettore è certamente interessante (tant'è vero che lo pubblichiamo!) ma presenta alcune involontarie pecche, che qui di seguito elenchiamo.

Approfittiamo dell'occasione, anzi, per far capire, ai nostri lettori, quali caratteristiche devono avere programmi di **uso universale**, soprattutto se in grado di attivare procedure **pericolosissime** quali la cancellazione di intere directory.

Anzitutto (benché non richiesto esplicitamente dalla "sfida") la procedura può esser migliorata considerando anche la particolare situazione che potrebbe veri-

```
.Key SourcePath,DirName,TargetPath

; "Muovi" - Scritto da Massimiliano Alessandri 1991
; Uso: EXECUTE MUOVI PathSorg. NomeDirectory PathDestin.
; Per informazioni consultare l'articolo esplicativo.

Copy C:Copy RAM:
CD RAM:
Copy C:IF IF
Copy C:EndIF RAM:
Copy C:Delete RAM:
Copy C:Echo RAM:
Copy C:Skip RAM:
Copy C:Lab Lab
FailAt 30
Copy >NIL: "<SourcePath><DirName>" RAM:BUFFER ALL
IF FAIL
Echo "*NNon ho caricate la dir. (mem. piena/err. disco)"
Skip Exit
EndIF

IF EXISTS "<TargetPath><DirName>"
Echo "*NLa directory esiste gia' su disco destin."
Skip Exit
EndIF

Copy >NIL: RAM:BUFFER "<TargetPath><DirName>" ALL
IF FAIL
Echo "*NNon ho trasferito la directory (errore su disco)"
Delete >NIL: "<TargetPath><DirName>" ALL
Skip Exit
EndIF

Delete >NIL: "<SourcePath><DirName>" ALL
Echo "*NProcedura riuscita !"

Lab Exit
Delete >NIL: BUFFER ALL
Delete >NIL: #?
```

Il semplice file batch per Amiga può essere migliorato

ficarsi in un caso già contemplato da AmigaDos.

E' noto, infatti, che il sistema operativo di Amiga consente, mediante una particolare forma sintattica del comando **Copy**, di spostare directory da un punto (leggi: directory) all'altro di uno stesso dischetto.

Si possono quindi, con il minimo sforzo, aggiungere un paio di righe al file batch, grazie alle quali la procedura sia in grado di "capire" se lo spostamento desiderato interessa due dischetti diversi oppure due directory dello stesso disco.

Si potrebbe obiettare che, in quest'ultimo caso, è inutile ricorrere al file batch dal momento che risulterebbe più semplice usare il comando Copy di AmigaDos.

Pur se l'osservazione è corretta, tuttavia, è bene ricordare che una procedura acquista maggiore interesse se il suo uso tende ad essere il più universale possibile: è inutile imparare a memoria la sintassi di **due** comandi (Copy e Muovi, nel nostro caso); è molto più semplice utilizzarne sempre e solo una; tanto più che il comando Copy di AmigaDos (a differenza del Copy dell'Ms - Dos) non è residente, ma occorre caricarlo tutte le volte che si intende usarlo.

Tanto vale, allora, attivare sempre e solo **Muovi** tutte le volte che occorre.

A parte le considerazioni sull'universalità della procedura, comunque, c'è da notare che nel caso particolare in cui, per distrazione, si tenta di effettuare un'operazione di muovi all'interno di uno stesso dischetto, lo spostamento avviene, ma con risultati un po' strani (si verifica una... somma di nomi!).

Se, però, quest'ultimo inconveniente può essere poco grave, conseguenze ben più letali comporta un'altra eventuale "distrazione".

Supponiamo, infatti, di usare due drive (DF0: e DF1) e di voler trasferire, ad esempio, la directory **Pippo** dal disco **Alfa:** al disco **Beta:**.

Se in quest'ultimo dischetto è già presente la directory Pippo (che contiene, magari, file del tutto diversi da quelli contenuti in **Alfa/Pippo**), l'omonima del disco Alfa non verrà cancellata (come è giusto che sia), ma quella già presente su Beta viene invece rimossa, con le conseguenze che potete immaginare!

Possiamo concludere, pertanto, che la sfida è ancora aperta e che, come tutte le altre sfide lanciate, non deve esser presa con leggerezza ma, al contrario, bisogna tener presente **tutti** i casi che, nell'uso frequente, possono capitarci.

E, dato che ci siete, pensate ad una più che rigorosa gestione del disco RAM: che, intelligentemente utilizzato nella procedura pubblicata, potrebbe creare difficoltà nel caso non riesca a contenere directory troppo lunghe.

In Ms - Dos

Ben diverso è il caso di **Maurizio F.** di Napoli che, ritenendo prevalente la tempestività rispetto alla correttezza della procedura, ha preferito inviare via **Fax** (e senza accordo preventivo) un file batch contenente addirittura alcuni errori di sintassi.

Per dimostrare che la fretta non paga, non pubblichiamo (ovviamente) la procedura inviata, invitando i lettori a scriverne una efficiente e, soprattutto, rispondente alle esigenze descritte.

Maurizio Zanello

Cara tastiera, quanto tempo è passato...

Stabiliamo il tempo trascorso in compagnia di un programma

Il presente articolo rappresenta la risposta alla sfida lanciata tempo fa da queste stesse pagine, in cui si richiedeva lo sviluppo di una procedura che stabilisse il tempo passato alla tastiera nell'uso di un certo programma.

Ad esempio, supponendo che un giorno usiamo un w/p per un'ora, il giorno successivo per un'ora e mezza ed il terzo giorno per un solo quarto d'ora, si chiedeva di scrivere un programma che, attivato in modo del tutto automatico tutte le volte che si lanciava quel determinato w/p, fosse in grado di indicare, nell'esempio accennato, il tempo di 2 ore e 45 minuti.

Sul N. 84 avevamo pubblicato la risposta relativa all'uso con l'**Amiga**. Stavolta la procedura riguarda un qualsiasi computer **Ms - Dos compatibile**.

Il comando DATE (così come pure TIME) agisce in modo piuttosto diverso in Ms - Dos rispetto all'AmigaDOS.

Infatti, mentre in quest'ultimo si limita a riportare la data e l'ora attuali, sul PC richiede *anche* l'immissione della nuova data (o ora).

Poichè questo risvolto può apparire talvolta seccante (come nel caso della redirezione, di cui si fa uso nel batch per Amiga pubblicato sul numero 84) è opportuno agire diversamente.

Sfruttando l'ottimo interprete/compilatore **QuickBasic** Microsoft, si possono seguire almeno due strade diverse.

La prima consiste nel realizzare due programmi, di cui il primo, brevissimo, avrebbe dovuto essere pressapoco così:

```
PRINT DATE$; " "; TIME$
```

...mentre il secondo poteva essere ricavato, con opportune modifiche, da quello scritto da Mirko Lalli in AmigaBasic.

In questo modo si otteneva, con il primo programma (compilandolo con il nome SHOWDATE, per esempio), un modo di funzionamento analogo a quello in AmigaDOS, e quindi si poteva anche utilizzare un batch con una redirezione simile.

Guida all'utilizzo

Ho invece preferito seguire una seconda strada, realizzando un **unico programma** in QuickBasic da lanciare prima e dopo l'esecuzione del programma principale.

In tal modo, infatti, risulta sufficiente scrivere un batch come questo che segue, che fa partire, ad esempio, il wordprocessor **WordStar**...

```
CD \WORDSTAR
TIMER
WS
TIMER
CD \
```

...per avere un processo completamente automatico, al termine del quale verrà visualizzato sullo schermo il tempo impiegato nell'utilizzo del programma, sia in modo **relativo** (cioè riferito unicamente all'ultimo uso) sia in modo **globale** (cioè da quando è stato applicato il "temporizzatore").

Nell'esempio riportato (ma anche in altre applicazioni), per maggior sicurezza risulta più opportuno inserire un altro CD \WORDSTAR immediatamente prima del secondo TIMER, in quanto WordStar prevede la possibilità di cambiare la directo-

```
'            **                    TIMER  v1.01                    **
'            **           by   Maurizio Zanello 1991               **
'            **   Per  PC MS-DOS  con  QuickBASIC Microsoft        **
ON ERROR GOTO errore
'  Imposta il nome del file dati.
td$ = "TIMER.DAT"
'  Fissa l'istante di inizio o di fine lavoro.
tm$ = LEFT$(TIME$, 5)
'  Controlla la validita' del file dati.
OPEN td$ FOR INPUT AS #1
INPUT #1, st$
IF st$ = "TIMER OFF" OR st$ = "TIMER ON" THEN
   '  Se il file dati e' valido allora leggi il suo contenuto.
   INPUT #1, tot$, ini$
   CLOSE #1
   '  Controlla lo stato del Timer.
   IF st$ = "TIMER OFF" THEN
   'Se il timer era spento, accendilo e salva l'ora di inizio.
      OPEN td$ FOR OUTPUT AS #1
      PRINT #1, "TIMER ON"
```

```
        PRINT #1, tot$
        PRINT #1, tm$
        CLOSE #1
        PRINT "Timer acceso"
    ELSE
        '  Se il Timer era acceso, spegnilo e calcola e salva
        '  il tempo di utilizzo.
        tini = VAL(LEFT$(ini$, 2)) * 60 + VAL(RIGHT$(ini$, 2))
        tfin = VAL(LEFT$(tm$, 2)) * 60 + VAL(RIGHT$(tm$, 2))
        tpar = tfin - tini
        'Controlla se c'e' stato il "passaggio" della mezzanotte.
        IF tpar < 0 THEN tpar = tpar + 1440
        tot = VAL(tot$) + tpar
        OPEN td$ FOR OUTPUT AS #1
        PRINT #1, "TIMER OFF"
        PRINT #1, STR$(tot)
        PRINT #1, STR$(tpar)
        CLOSE #1
        '  Calcola il valore di tempo totale in ore e minuti.
        toth = INT(tot / 60)
        totm = tot - toth * 60
        '  Visualizza i tempi.
      PRINT "Il programma e' stato utilizzato per"; tpar; "min."
      PRINT "Tempo utilizzo tot. e' di"; toth; "h e"; totm; "min."
    END IF
ELSE
 'Se file dati non e' valido avvisa che non puo' funzionare.
    CLOSE #1
    BEEP
   PRINT "Conflitto di programm: Il file "; td$; " e' stato gia'"
   PRINT "creato precedentemente per una diversa applicazione."
   PRINT "E' necessario cancellarlo o ridenominarlo."
END IF
END
'  Controllo errori.
errore:
IF ERR = 53 THEN
 'Se file dati non e' presente crealo, azzera Timer, ricomincia.
    OPEN td$ FOR OUTPUT AS #1
    PRINT #1, "TIMER OFF"
    PRINT #1, "0"
    PRINT #1, "0"
    CLOSE #1
    RESUME
ELSE
 'Se c'e' altro errore, mostrane il num. corrisp. e fermati.
    PRINT "Errore n."; ERR
    ON ERROR GOTO 0
END IF
```

ry corrente, e questo può portare il temporizzatore a lavorare con un altro file-dati eventualmente presente su un'altra dir.

Il programma in QuickBasic

Il listato è piuttosto semplice da comprendere e da digitare e una volta compilato occupa all'incirca **38**Kb. Per il suo modo di funzionare può essere definito un "programma interruttore", dal momento che, lanciato una seconda volta, si comporta in maniera "simmetrica" rispetto a quando è stato fatto partire la prima volta.

Dato, inoltre, che effettuerà operazioni di **I/O** su disco, vi è inserita una semplice routine di controllo degli errori, che si incarica, tra l'altro, di creare il file dati qualora non fosse presente nella directory attuale (e cioè ogni volta che il programma TIMER viene usato per la prima volta).

All'inizio dell'esecuzione il programma imposta il nome del file dati (modificabile a piacere nel listato sorgente) e fissa l'istante di tempo con la precisione di un minuto.

Successivamente apre in lettura il presunto file dati e controlla se è tale prelevando la **prima stringa** ivi contenuta.

Se il file dati non è quello richiesto (quando, ad esempio, è stato scritto per altri scopi), il programma chiude il file, comunica all'utente che non può svolgere il suo compito e si ferma, elencando possibili soluzioni.

Se, invece, il file dati è quello giusto, preleva gli altri valori contenuti (rispettivamente il tempo di utilizzo totale espresso in minuti e l'orario di inizio o il tempo di utilizzo relativo) e lo chiude.

Poi controlla lo stato attuale (memorizzato nella prima stringa prelevata) del file e...

* Nel caso fosse *"spento"*, apre il file dati in scrittura e vi scrive il nuovo **stato**, il tempo **totale** e l'orario di **inizio**, richiude il file e termina...

* Nel caso fosse *"acceso"*, calcola il tempo di utilizzo **relativo**, lo somma a quello totale, apre il file dati in scrittura, scrive nel file il nuovo stato (cioè "spento"), il tempo totale e quello parziale (sempre in minuti), richiude il file, visualizza i tempi ottenuti (il tempo relativo solo in minuti, quello totale in ore e minuti), e termina.

Alcune precisazioni

Il calcolo dei tempi viene svolto in **minuti**, anziché in **ore**: abbiate pietà, sono ancora sotto l'influenza delle imprecisioni del C/64 verso i numeri con la virgola.

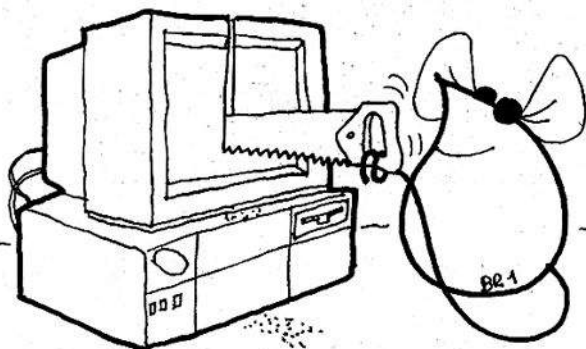
Il programma è in grado di accorgersi se c'è stato il "passaggio della mezzanotte", cioè evita il paradosso di aver lavorato per **-10** minuti quando lo si usa ad esempio tra le 23:55 e le 0:05.

Abbiate l'accortezza di accertarvi che il file dati non esista o sia "spento" prima di lanciare il batch dell'esempio.

Per sapere il tempo di utilizzo totale, in ogni momento è sufficiente impartire un **TYPE TIMER.DAT** per vedere visualizzato lo stato del file, il tempo totale (espresso in minuti) e il tempo relativo (o l'orario di inizio).

di Andrea Preziosi

Alla scoperta dei file compattati

Tutti noi, abitualmente, compattiamo i file, soprattutto se lunghi; ecco due utility in Basic da usare con l'arcinoto PkUnzip

Il programma di queste pagine può essere utilizzato solo se si possiede il programma di scompattazione **Pkunzip**, noto almeno quanto il corrispondente compattatore **PkZip**.

Chi possiede altri compattatori o altri computer, tuttavia, può adattare i presenti listati con la massima facilità.

Supponete di aver registrato su alcuni dischetti vari file e directory in formato compresso; supponiamo, inoltre, di dover rintracciare un file, ma non ricordate esattamente in quale file compattato esso si trova.

Decompattare tutti i file, per risolvere il problema, non é certo la soluzione più pratica perché richiede troppo tempo; in questi casi, di solito, si usa l'opzione **-v** di PkUnzip, operazione che consente un considerevole risparmio di tempo; purtroppo la leggibilità dell'output generato è discutibile, specie se le righe stampate superano la larghezza dello schermo e vengono visualizzate su due righe successive, perdendo l'utile incolonnamento.

```
PKUNZIP (R)    FAST!    Extract Utility    Version 1.1    03-15-90
Copr. 1989-1990 PKWARE Inc. All Rights Reserved. PKUNZIP/h for help
PKUNZIP Reg. U.S. Pat. and Tm. Off.

Searching ZIP: b:PKDIR.ZIP

 Length  Method   Size  Ratio   Date     Time   CRC-32   Attr  Name
 ------  ------  ----- -----   ----     ----   ------    ----  ----
   4352  Implode   2228  49%  23-06-91  17:46  c3268e65  --w  PKDIR/PKDIR
   4352  Implode   2228  49%  23-06-91  17:46  c3268e65  --w  PKDIR
   1291  Implode    829  36%  06-06-91  15:26  27624167  --w  ORIGINAL/PKDIR1.BAS
   1329  Implode    758  43%  06-06-91  15:22  d0e2ff3b  --w  ORIGINAL/PKDIR.BAS
   1536  Implode    855  45%  23-06-91  17:49  8b32f731  --w  TRATTATI.WS/PKDIR1_1.BAS
   1536  Implode    782  50%  23-06-91  17:50  b3023fba  --w  TRATTATI.WS/PKDIR_1.BAS
 ------          ------  ---                                  -------
  14396            7680  47%                                     6
```

Le informazioni che compaiono con la sintassi -V PkUnzip...

```
PKDIR/PKDIR
PKDIR
ORIGINAL/PKDIR1.BAS
         PKDIR.BAS
TRATTATI.WS/PKDIR1_1.BAS
            PKDIR_1.BAS
```

...e quelle visualizzate con PkDir !

```
1 Rem Gw Basic
10 OPTION BASE 1: DEFINT A-Z
20 DIM A$(100), B$(100)
30 CLS : KEY OFF
40 LINE INPUT "NOME DEL FILE DA ESAMINARE:", NF$
50 IF NF$ = "" THEN END
60 LINE INPUT "VUOI LA STAMPA SU CARTA (S/N):", DEV$
70 ' crea file da leggere
80 S$ = "pkunzip " + NF$ + " -v > pkdir"
90 SHELL S$
100 ' apre file da leggere
110 OPEN "i", #1, "pkdir"
120 ' salta le linee inutili
130 FOR X = 1 TO 9: LINE INPUT #1, A$: NEXT
140 ' inizia a leggere le linee utili
150 LINE INPUT #1, A$
160 IF LEFT$(A$, 7) = " ------" THEN 200
170 I = I + 1
180 A$(I) = RIGHT$(A$, LEN(A$) - 61)
190 GOTO 150
200 CLOSE #1
210 ' cancella il file letto
220 KILL "pkdir"
230 'elimina alcune subdirectory per consentire max leggibilità
240 B$(1) = A$(1)
250   FOR X = 2 TO I
260   CE = 0
270   Z = INSTR(CE + 1, A$(X), "/")
280   IF Z = 0 OR Z > LEN(A$(X - 1)) THEN 310
290   IF LEFT$(A$(X),Z) = LEFT$(A$(X - 1),Z)THEN CE = Z ELSE 310
300   GOTO 270
310   B$(X) = SPACE$(CE) + RIGHT$(A$(X), LEN(A$(X)) - CE)
320 NEXT
330 ' stampa l'output sulla periferica desiderata
340 IF DEV$ = "S" OR DEV$ = "s" THEN FOR X = 1 TO I:
    LPRINT B$(X): NEXT: GOTO 40
350   FOR X = 1 TO I
360   PRINT B$(X)
370   IF X / 22 = INT(X / 22) THEN PRINT "premi un tasto":
    A$ = INPUT$(1)
380   NEXT
390 GOTO 40
```

La versione Gw - Basic

PkDir

Con il programma descritto in queste pagine (nel seguito, **PkDir**), é possibile avere un output limitato allo stretto indispensabile dal momento che viene visualizzata una struttura ad albero simile a quella nota in vari Tools, solo che mancano le linee di separazione (tipiche dell'output di PkUnzip) ma sono presenti solo i nomi dei file (vedi figure).

Esteticamente può lasciare a desiderare, ma in compenso offre un'ottima leggibilità.

L'algoritmo è riportato in due linguaggi, che esaminiamo separatamente.

La versione in **QuickBasic** può funzionare solo se compilata e se riceve i parametri direttamente dal **Dos**.

Supponendo di voler conoscere i file (e le directory) contenuti all'interno del file **a.zip**, dovremo impartire, da Dos, il comando...

```
pkdir a.zip
```

...e volendo la stampa su carta, invece...

```
pkdir a.zip /p
```

La versione in **GwBasic** può funzionare anche interpretata, ma é meglio compilarla per ottenere una maggiore velocità di esecuzione.

Il programma chiede, all'inizio, il nome del file da esaminare e se si desidera anche l'output su stampante.

Dopo aver visualizzato l'output, il programma chiede il nome di un altro file da visualizzare; se non ve ne sono altri basta premere il tasto Return.

Il principio di funzionamento é identico per entrambe le versioni: tramite il comando **Shell** viene lanciato, da Dos, il programma **PkUnzip** (che, quindi, deve esser presente nella directory di "lancio") con l'opzione **-v**; l'output viene diretto verso un file (di nome **PkDir**, eventualmente modificabile dal lettore; suggeriamo di usare, ad esempio, l'inconsueto nome **xxxxxxxx.xxx** per evitare inopportune sovrapposizioni, magari con lo stesso programma Basic!) che, alla fine del trasferimento, conterrà esattamente ciò che sarebbe apparso su video se aves-

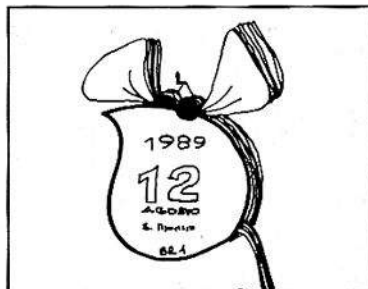
```
REM Versione Quick Basic
DIM a$(1 TO 100), b$(1 TO 100)
DEFINT A-Z
' controlla i parametri
IF COMMAND$ = "" THEN PRINT "MANCANO I PARAMETRI": END
IF RIGHT$(COMMAND$, 3) = " /P" THEN
                dev$ = "PRN"
                nf$ = LEFT$(COMMAND$, LEN(COMMAND$) - 3)
        ELSE
                nf$ = COMMAND$:
                dev$ = "CON"
END IF
' crea file pkdir con ridirezione dell'output
s$ = "pkunzip " + nf$ + " -v > pkdir":
   REM Cambiare eventualmente il nome al file da redirigere!
SHELL s$
' apre il file
OPEN "i", #1, "pkdir": REM Cambiare eventualmente nome e path
' salta le linee inutili
FOR x = 1 TO 9: LINE INPUT #1, a$: NEXT
' legge le linee contenenti informazioni sulle subdirectory
  DO
  LINE INPUT #1, a$
  IF LEFT$(a$, 7) = " ------" THEN EXIT DO
  i = i + 1
  a$(i) = RIGHT$(a$, LEN(a$) - 61)
  LOOP
CLOSE #1
' cancella file pkdir
KILL "pkdir": REM Attenzione ad usare un nome appropriato!!
' elimina alcune subdirectory per consentire la max leggibilità
b$(1) = a$(1)
  FOR x = 2 TO i
  ce = 0
    DO
    z = INSTR(ce + 1, a$(x), "/")
    IF z = 0 OR z > LEN(a$(x - 1)) THEN EXIT DO
IF LEFT$(a$(x),z) = LEFT$(a$(x - 1), z) THEN ce = z ELSE EXIT DO
    LOOP
  b$(x) = SPACE$(ce) + RIGHT$(a$(x), LEN(a$(x)) - ce)
  NEXT
' stampa l'output sulla periferica desiderata
IF dev$ = "PRN" THEN FOR x = 1 TO i: LPRINT b$(x): NEXT: END
  FOR x = 1 TO i
  PRINT b$(x)
  IF x MOD 22 = 0 THEN PRINT "premi un tasto": SLEEP
  NEXT
```

La versione QuickBasic

simo lanciato PkUnzip nel modo consueto.

Qualunque sia il numero di file e di subdirectory contenuti nel file compattato, il file PkDir (o xxxxxxxx.xxx) seguirà sempre uno standard definito: le prime 9 linee contengono sempre, infatti, informazioni su PkUnzip (che il programma PkDir provvede a cancellare perchè non servono al nostro scopo), sui file (numero di byte iniziali e finali, data, eccetera) e, a partire dal **61**imo carattere, il path completo di ciascun file compattato; subito dopo è presente una linea formata da numerosi trattini ed, infine, una linea contenente i resoconti totali.

Il programma PkDir legge, dal file rediretto, **solo** le informazioni che servono, cancellandolo subito dopo; i path completi di ogni file vengono memorizzati nel **vettore A$**, quindi il programma li esamina uno alla volta e cancella tutte le subdirectory eventualmente già ripetute. I messaggi così "depurati" vengono quindi registrati nel **vettore B$**, che verrà poi riportato su schermo o su carta in base alle richieste dell'utente.

Il programma può elaborare file **.zip** in cui sono stati compattati fino ad un massimo di **100 file**. Per compattazioni di maggior respiro é sufficiente modificare l'indice dei vettori indicato all'inizio di ogni versione del programma.

Tempo di quiz, 22 domande per rinfrescare la memoria

Dato il periodo vacanziero, riteniamo che molti dei nostri affezionati lettori apprezzeranno la possibilità di rispondere ai classici quiz enigmistici da "ombrellone", rispolverando nozioni apprese dalla nostra rivista, con qualche sorriso.

☞ 1) La IBM nacque:

a) Nel 1936, da Jan Gorlan quando cambiò il nome alla NCR
b) Nel 1924, da Thomas Watson quando cambiò il nome alla CTR
c) Nel 1969, da Steve Jobs che cambiò il nome alla Citron

☞ 2) Le stampanti Ink Jet funzionano tipicamente:

a) Inviando un raggio laser direttamente sul foglio

b) Con dei microgetti di inchiostro

c) Con un raggio laser che pilota un getto di inchiostro

☞ 3) Un benchmark è precisamente:

a) Un banco di prova per computer guasti

b) Un segnale che indica all'operatore quando può sedersi
c) Un sistema per misurare le prestazioni del computer

☞ 4) La velocità massima raggiunta dai più moderni modem disponibili secondo lo standard Bell/CCITT con normali linee telefoniche vocali è di:

a) Circa 1200 bit al secondo
b) Circa 2400 bit al secondo
c) Circa 14.400 bit al secondo

☞ 5) La parola Modem significa:

a) Modulator Demodulator
b) Monitor and Demodulator
c) "ora ti morsico" in romanesco

☞ 6) Il pacchetto Windows consente di lavorare in pieno multitasking, ma:

a) Solo possedendo un 80386/486
b) Solo possedendo un 80486
c) Accettabilmente con pochi task

☞ 7) La parola "Postscript" rappresenta:

a) Una opzione di videoscritture per scrivere domani quello che si potrebbe scrivere oggi.
b) Un linguaggio di programmazione come il basic realizzato della Adobe Inc.
c) Un linguaggio di definizione di pagine da stampare

☞ 8) Il sistema operativo MS-DOS V1.00 fu scritto:

a) Interamente in linguaggio macchina
b) In gran parte in C con piccole porzioni Assembler
c) In Cobol

☞ **9) La differenza principale tra un compilatore ed un interprete è che:**

a) Il compilatore lavora più velocemente dell'interprete
b) Il compilatore deve svolgere tutto il proprio lavoro prima di potere eseguire il programma
c) Il compilatore, dal tempo degli amanuensi, scrive di più e viene pagato di meno dell'interprete

☞ **10) Un programma Postscript è normalmente:**

a) Una lunga sequenza di comandi scritti in ASCII
b) Una serie di subroutine in linguaggio macchina
c) Un programma che consente di sfruttare i fonti di Amiga

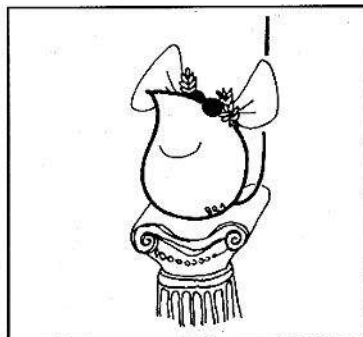
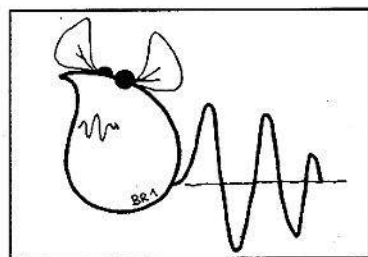
☞ **11) Il primo computer elettromeccanico della storia fu costruito:**

a) Da Blaise Pascal per facilitare compiti di cambio valuta
b) Da Thomas Watson, alla IBM, impiegando tre anni
c) Da Howard Aiken, ad Harvard, impiegando cinque anni

☞ **12) L'algebra di Boole è effettivamente:**

a) La libreria matematica dei linguaggi ad alto livello
b) Un insieme di regole per il trattamento logico di numeri binari
c) Un tipo di geometria non euclidea

☞ **13) I nuclei di ferrite sono:**

a) Un vecchio strumento per la costruzione di memorie
b) Anelli di fissaggio diamagnetici per meccaniche di floppy disk drive

c) L'equivalente dei nostri globuli rossi per i moderni automi usati nelle catene di montaggio della FIAT

☞ **14) L'istruzione di chiamata all'interruzione BIOS 1AH disponibile a partire dall'IBM AT:**

a) Permette la scrittura di un carattere ASCII sul video
b) Non è standardizzata e non lo dice nemmeno Giancarlo Mariani nel suo corso a che cosa serve
c) Permette la lettura e scrittura di data ed ora correnti

☞ **15) Il linguaggio Fortran nacque**

a) In una notte buia e tempestosa alla AT&T
b) Nel 1958 alla Digital
c) Tra il 1954 ed il 1957 all'IBM

☞ **16) La differenza tra i due pacchetti per Ms - Dos chiamati Borland C++ e Turbo C++ è che:**

a) Il primo è una versione semplificata del secondo
b) Il primo è una versione resa Windows-compatibile del secondo
c) Il primo è della Borland ed il secondo della Microsoft

☞ **17) Una Tenbyte è effettivamente:**

a) Quella sporca decina di byte
b) Un gruppo di dieci byte secondo vari linguaggi assembler
c) Un gruppo di dieci byte secondo l'Assembler 80286

☞ **18) Per produrre la risoluzione di 1024 x 768 con 256 colori una scheda Ultra-VGA deve possedere:**

a) 512K di memoria montata
b) 1M byte di memoria montata
c) Quattro cucchiai di panna montata

☞ **19) Una "variabile" è materialmente:**

a) Una isobara difficile da interpretare per i metereologi
b) Una zona di memoria alterabile dal programma
c) Un tipo di dato il cui significato varia in funzione di eventi esterni all'elaboratore

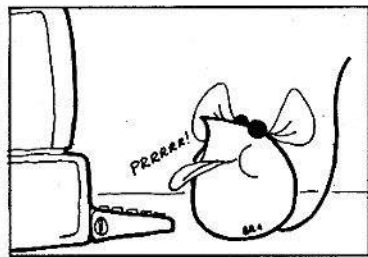
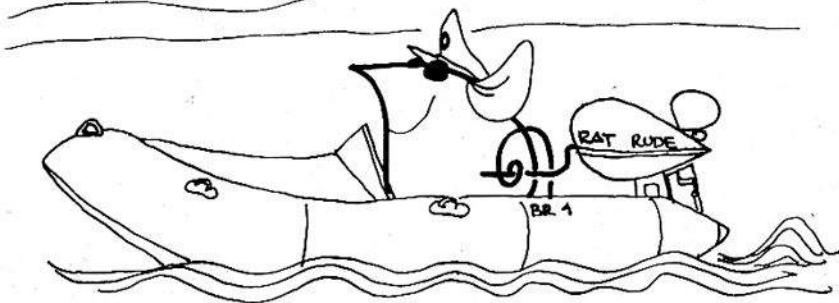
☞ **20) Rovinando i dati di gestione della struttura del disco su di un disco rigido in AmigaDOS si ottiene:**

a) Un disco i cui file sono disposti casualmente
b) Un disco recuperabile teoricamente anche completamente
c) Un disco recuperabile con estrema difficoltà

☞ **21) Quali delle seguenti sequenze comprende una parola che non c'entra con le altre:**

a) Open, Print, Let, Option, For, Read
b) For, Sprintf, Long, Switch, Break
c) Movzx, Mul, Ins, Mov, Cwd

☞ **22) Il formato IFF della Electronic Arts, usato sotto vari sistemi operativi, prevede effettivamente**

a) Soltanto file grafici e sonori
b) File grafici, sonori, testuali
c) Dipende dalle implementazioni

Risposte al Quiz di questo numero

1) b. Considerato come la neonata IBM superò la grande crisi del 1939, possiamo stare certi che supererà anche l'attuale momento difficile!

2) b. Dove i microgetti possono essere pilotati con varie tecnologie, come ad esempio da un campo magnetico. La (a) indica le stampanti laser tipo.

3) c. Se avete risposto (b) la vosta dipendenza dal computer comincia a diventare patologica.

4) c. Con i modem V32bis e V42bis. La (a) è quanto credono e sostengono tuttora molti dirigenti SIP, che chiamano ad "alta velocità", appunto, i vecchi modem a 1200 baud.

5) a. Detto ampiamente in vari articoli sui modem di CCC.

6) c. In ogni caso, per avere una velocità accettabile, è importante possedere un 80386/486 sufficientemente velocel.

In caso contrario, infatti, con più di due task il sistema diventa insopportabilmente lento (lo diventa spesso comunque!).

7) c. Sebbene insidiato da vari contendenti, il Postscript rimane tuttora il più diffuso sistema di definizione delle stampe in sistemi professionali.

8) b. Se avete risposto C non sapete che cosa è il Cobol!

9) b. La (a) è fuorviante, in quanto è il codice compilato che viene eseguito più velocemente, non il compilatore che lavora più velocemente dell'intereprete.

10) a. Molti dossisti possono avere risposto (c), ma esistono varie riviste nel mondo prodotte in PostScript con sistemi Amiga, a costi più bassi di quanto possibile con Macintosh.

11) c. Il "computer" di Pascal era meccanico, senza ovviamente parti elettriche.

12) b. Se avete risposto (a), in una scala di Quoziente Intellettivo da 0 a 130 voi siete a -5...

13) a. Usati nei vecchi elaboratori elettromeccanici insieme a schede perforate ed altri strumenti antiquati.

14) c. Non è possibile che il Mariani non dica qualcosa nel suo corso, suvvia!

15) c. Può trarre in inganno la data variabile, ma fu effettivamente un parto difficile. Se avete risposto (a) vi confondete con i fumetti di Snoopy.

16) b. Se la Microsoft sa che avete risposto (c), vi manda a casa una copia del suo Ms - Dos V4.00 (bacatissimo), così imparate...

17) c. Tale definizione compare in riferimento all'80286 appunto per indicare un particolare tipo di dato trattabile direttamente dalla CPU.

18). b. La (c) è tratta dalla ricetta per ottenere una buona Sachertorte e questa non è la rivista "Guida cucina"...

19) b. La (c) è accattivante ma parziale. La (a) è falsa perchè in caso di difficoltà i meteorologi ci dicono "tempo variabile" ed il problema è risolto.

20) b. Uno dei motivi per cui AmigaDOS è così lento è perchè deriva appunto da un sistema di filing per grossi elaboratori (TRIPOS) molto sicuro, ma studiato per dischi di grande capacità e velocità, purtroppo!

21) c. Molto cattiva come domanda: la "movzx" è riservata al solo 80386 mentre le altre sono per tutti i processori Intel 80x86.

22) c. Sebbene molti possano avere risposto (quasi) correttamente (b), è da notare che il formato è in continua evoluzione.

Questa è in funzione dell'evolversi delle caratteristiche del software e dell'hardware in circolazione.

Estate '91

A che punto siamo? (indagine tra i lettori)

Riproponiamo anche questo mese, ai lettori che non hanno aderito all'invito del numero scorso, la possibilità di esprimere il loro giudizio sulla nostra rivista: tutto ciò che volete comunicarci, quantomeno le opinioni più importanti, potete farle conoscere compilando il semplice questionario di queste due pagine; Non abbiate timore a staccare il foglio: seguendo la linea tratteggiata eviterete che la pagina "opposta" si distacchi dal fascicolo. Chi lo desidera, ovviamente, potrà inviare la fotocopia riproducente l'inchiesta.

Domanda N. 1 (esperienze precedenti)

Da quale esperienza informatica provieni?

- ❑ Autodidatta (hobby, club, amici, riviste, ecc.)
- ❑ Ho seguito un corso (scuola, università, corrispond.)
- ❑ Lavoro nel settore specifico

Dom. N. 2 (computer usati IN PASSATO)

Quali computer hai usato in precedenza?

- ❑ Il C/64 (oppure Vic 20, C/128, C/16, +4, Pet)
- ❑ Lo Spectrum (oppure Msx, QL, Apple 6502)
- ❑ Amiga oppure Ms - Dos

Dom. N. 3 (computer usati OGGI)

Quali computer usi abitualmente?

- ❑ Quello/i indicato/i nella domanda N.2
- ❑ Amiga
- ❑ Ms - Dos

Dom. N. 4 (computer che USERAI)

Quale computer pensi di usare nel corso del 1992?

- ❑ Quello/i indicato/i nelle dom. 2 e/o 3
- ❑ Amiga
- ❑ Ms - Dos

Dati del lettore (compilazione facoltativa)

Cognome

Nome

Indirizzo

(CAP) Città **Tel.**

Domanda N. 5 (configurazione)

Quanti drive ha il sistema indicato nella dom. n. 4?

- ❑ 1 da 5.25
- ❑ 2 (o più) da 5.25
- ❑ 1 da 3.5
- ❑ 2 (o più) da 3.5
- ❑ Hard disk N. Megab. (specificare)

Domanda N. 6 (memoria RAM)

Di quanta RAM dispone il sistema (ind. in dom. n. 4)?

- ❑ 500 K(se Amiga) oppure 640 K(se Ms - Dos)
- ❑ 1 mega
- ❑ N. megabyte (specificare):

Domanda N. 7 (stampante)

Quale stampante userai con il sistema di dom. 4?

- ❑ Un modello a 9 aghi (oppure nessuna)
- ❑ Un modello a 24 aghi (o comunque di medio prezzo)
- ❑ Una laser (o un modello di prezzo elevato)

Domanda N. 8 (modem)

Nel 1992 disporrai di un modem?

- ❑ No
- ❑ Sì, un modello medio (prezzo fino a 500 mila lire)
- ❑ Sì, un modello di elevate prestazioni (oltre le 500 mila)

Dom. N. 9 (prezzo della rivista)

Saresti disposto all'aumento del prezzo di copertina se decidessimo di inserire un dischetto (ovviamente del formato valido per il computer indic. in dom. 4)?

- ❑ No, in nessun caso il prezzo deve aumentare
- ❑ Sì, purchè contenga *anche* giochi e simili
- ❑ Sì, purchè contenga *anche* s/w di tipo professionale
- ❑ Sì, purchè contenga *anche* utility
- ❑ Sì, purchè il prezzo di copertina non risulti mai superiore a L. (indicare cifra max.)

N.B: è possibile barrare più risposte, purchè non in contraddizione tra di loro

Inserire in busta chiusa, affrancare e spedire a:

Systems Editoriale
Via Mosè 22
cap 20090 Opera (Mi)

Dom. N. 10 (Argomenti preferiti)

A quali argomenti vorresti si dedicasse più spazio?

- ❑ Recensioni di software e/o hardware
- ❑ Didattica (linguaggi C, Basic, Pascal, Assembly, ecc.)
- ❑ Uso ottimale di programmi ed utility commercializzati (Word processor, DTP, Spreadsheet, Database, ecc.)

Un'importante precisazione

Prima di rispondere alle domande indicate, ricorda che molte di queste si riferiscono al sistema computerizzato indicato nella **domanda N.4** (quello, cioè, che ritieni di utilizzare in prevalenza nel corso del prossimo anno **1992**). Se ritieni che continuerai ad usare il computer che usi tuttora (segnalato nella domanda **n. 3**), ovviamente, le risposte che fornirai dovranno riferirsi a quest'ultimo.

di Marcello Magnifico

Pi greco, parte seconda

Un paio di righe, addirittura in semplice Basic, ed ecco che il magico numero pigreco viene fuori dal vostro computer

E' scritto sul numero **83** di C.C. che il metodo per calcolare pi greco tramite una serie del tipo...

a (n) = a (n - 1) + 4/n

...è lentissimo, tanto che perfino dopo 16000 iterazioni il valore fornito dal lumacoso Commodore 64 è lontano da quello noto:

π= 3.141592653589793238462643...

Ci si chiedeva se, oltre alla lentezza tipica della procedura (ed all'hardware usato, il modesto microprocessore ad otto bit che caratterizza il C/64) vi fossero altre cause che limitassero la velocità di esecuzione di un simile calcolo. Viene quindi presentato un altro algoritmo, universalmente noto presso gli studiosi di matematica, ma per molti completamente nuovo.

L'idea è venuta scrivendo un programma banale in cui veniva fatta eseguire, al calcolatore, sempre la stessa operazione su di una variabile; di questa, in seguito, veniva scritto il contenuto sullo schermo (in modo simile ad una serie). Facendo ricorso ad alcune funzioni trigonometriche si arrivava ad approssimare una costante tipica dell'equazione impostata.

Trovando la cosa molto interessante, e continuando con altre funzioni, si pervenne ad un programma di questo tipo, digitato sempre sul C/64:

```
 0 X = .1
10 X = X + Sin(x): Print X
20 If X1 = X Then End
30 X1 = X: Goto 10
```

Inserire la linea 0 era stato necessario in quanto **Sin(0) = 0**; in caso contrario non si otterrebbe altro che degli zeri.

Dopo il Run si può notare qualcosa di importante, come si può notare nella tabella riportata a fine paragrafo

Il programma si ferma proprio sul "pi greco", e dopo sole 9 iterazioni, non 16000...

La spiegazione del fenomeno si può individuare pensando al fatto che il computer calcola il seno a partire da un argomento specificato in **radianti** (il GFA-Basic del Commodore Amiga permette anche la conversione in gradi, e viceversa).

Accade, quindi, che Sin (pi_greco) = 0, come sa chi ha studiato trigonometria, e che avvicinandosi sempre più al pi greco il valore del seno diminuisce fino a diventare nullo. Notiamo, comunque, che il C/64 colpisce ancora: viene visualizzato un valore di **3.14159266** contro quello di **3.14159265** dato dal tasto apposito.

Non resta che da chiederci perchè mai la Commodore non abbia provveduto a riscrivere le routines matematiche in modo da renderle più affidabili...

Il programma per Amiga

E' scritto in **Amigabasic** e sfrutta in pieno lo splendore della doppia precisione (**15** cifre!). Non ha bisogno di commenti, a parte uno: se lo si vuole rendere più veloce è sufficiente sostituire l'**1** della prima linea con un **3**, o magari con **3.14**. Funzionerà lo stesso, e meglio.

```
.199833417     3.09665149
.398339482     3.14157753
.78622785      3.14159266
1.49392106     3.14159266
2.49096762     READY.
```

Ultima precisazione

L'autore del programma pubblicato sul n. 83 lamentava (giustamente) la restrizione del calcolo alla precisione del calcolatore stesso.

Resta da dire che riscrivere gli algoritmi di calcolo (anche usando un linguaggio evoluto) non è affatto cosa breve e semplice, al contrario del rapido algoritmo qui pubblicato. Inoltre, dato che i computer sono stati ideati per prima cosa come "calcolatori", non avrebbe senso calcolare il pi greco a 100 decimali se poi non ci si trovasse nelle condizioni di poterlo sfruttare interamente.

Rimane solo la curiosità, ma per quella si perde meno tempo a cercare su di una tabella...

Per curiosità, ricordiamo che per implementare le funzioni trigonometriche su di un computer è necessario ricorrere ad una formula, detta di **Taylor**, che permette l'approssimazione di una qualunque funzione nell'intorno di un suo punto, con errori che possono essere resi piccoli a piacere.

```
CLS: z#=0: x#=1: REM Mini - micro programma in AmigaBasic per calcolare Pi Greco
loop:
x# = x# + CDBL (SIN (x#)): IF x#  z# THEN z# = x#: GOTO loop: ELSE PRINT x#: END
```

Una sfida al mese

Ok, il prezzo è giusto!

Quanto vale il nostro vecchio computer? Scriviamo insieme una procedura per determinare il valore di un apparecchio usato

Questo mese la sfida non richiede necessariamente l'uso di un computer: una calcolatrice tascabile (anche di quelle che, ormai, si trovano nei fustini di detersivo) può essere adatta allo scopo.

Se, poi, chi legge queste righe si trova sotto l'ombrellone (o all'ombra di una pineta), può tentare di eseguire alcuni cacoli a mente (o con carta e penna) ripromettendosi, tornato in albergo (o al ritorno dalle ferie) di impegnarsi più attentamente. Chi smanetta con i computer, e chi è appassionato di elettronica in generale, è certamente conscio del fatto di essersi imbattuto in un hobby che, al contrario di tutti gli altri, richiede una quantità di denaro costantemente minore a mano a mano che il tempo trascorre.

Chi, ad esempio, ama usare la motocicletta, sa benissimo che i prezzi di listino delle moto aumentano anno dopo anno; spendendo cinque milioni oggi per l'acquisto di un mezzo, la stessa moto (o meglio, un suo modello equivalente, dal momento che vengono apportate continue modifiche estetiche e meccaniche) costerà almeno in 50% in più tra quattro anni. Questo fatto consente, ad occhio e croce, di rivendere la moto che oggi acquistiamo, tra quattro anni, ad un prezzo relativamente alto se confrontato con quanto abbiamo speso per procurarcela.

Uno dei progenitori degli attuali Laptop: quanto può valere?

Nel campo dei **TV color** o dei videoregistratori, passando ad altro settore merceologico, il costante abbassamento dei prezzi e le notevoli migliorie apportate, consentono di riempire le colonne degli annunci economici di una quantità incredibile di apparecchi, ancora validi, a prezzi stracciati.

Computer

Computer ed accessori, ahinoi, seguono una sorte ancora peggiore del mercato dei TV color; questi, infatti, possono esser valutati in base alla presenza del **Televideo** o della ricezione **stereo** (il telecomando, infatti, ormai è in dotazione standard di quasi tutti i modelli); poco importano, nel mercato dell'usato, le caratteristiche "interne" di un TV color: se una certa funzione viene svolta da una scheda a transistor piuttosto che dalla presenza di un processore più moderno, il telespettatore non se ne accorge, dal momento che la funzione viene svolta in modo "trasparente".

Nel campo dei computer, invece, la differenza c'è, e si vede.

Se, ad esempio, il processore è un **8088** oppure un **80286**; se il drive montato è da 1.44 mega oppure da 720 K; se la scheda video è una CGA oppure una super VGA; in tutti questi casi, insomma, le caratteristiche sono nettamente evidenti anche per i più sprovveduti acquirenti.

Valutazioni

Sorge, quindi, un problema: come valutare obiettivamente un apparecchio? Come indicare con precisone il valore di un accessorio, in un mercato in rapidissima evoluzione tecnologica, che non può avere, come indice di riferimento, una tabella del tipo simile a quelle pubblicate da **Quattroruote** o da **Gente Motori**?

Un qualsiasi tentativo di tabellare prezzi, infatti, si scontra con la necessità di aggiornamenti continui, governati spesso da politiche commerciali aggressive e rapidissime che consentono di offrire apparecchi che, solo il mese precedente, erano valutati in maniera molto più generosa.

La sfida di questo mese, quindi, consiste nel determinare un procedimento sufficientemente inequivocabile e attivabile "in tempo reale" per stabilire, con un margine accettabile, il reale valore commerciale di un sistema computerizzato o di un semplice accessorio.

In queste pagine ci limiteremo a dare un suggerimento sul modo di sviluppare la procedura richiesta.

Modelli

Il valore di un computer usato non può prescindere dalla possibilità di reperire, al momento della sua valutazione, lo stesso modello nel "circuito" del nuovo, vale a dire presso rivenditori **autorizzati**.

Nel mese di ottobre 1991, ad esempio, il valore di una stampante laser commercializzata recentemente, ed acquistata nel mese di maggio 1991, avrà, percentualmente, un valore maggiore rispetto ad un computer AT basato su un microprocessore 80286 prodotto nel 1987.

La commercializzazione della stampante laser cui ci riferiamo, infatti, ha avuto inizio da poco tempo; nel mese di ottobre '91, quindi, non avrà ancora subìto quelle modifiche alle quali, periodicamente, vengono sottoposti tutti gli apparecchi.

Il computer AT-286, invece, è **simile** all'analogo modello AT-286 reperibile presso i rivenditori autorizzati (ci riferiamo, ovviamente, sempre al mese di ottobre '91), ma presenterà caratteristiche che lo pongono in secondo piano rispetto al modello di produzione attuale (clock 8/10 Mhz rispetto alla maggiore velocità oggi disponibile; schede hardware con integrazione diversa dei componenti, eventuale difficoltà di reperire parti di ricambio, eccetera).

Riepiloghiamo: per stabilire il valore di un apparecchio nel mese di ottobre 1991, bisognerà tener conto del prezzo di vendita al pubblico del medesimo apparecchio; se il modello che interessa non è più presente sul mercato, bisognerà tenerne conto (la "moda" ha un prezzo, anche nel campo dell'informatica).

Facciamo rilevare esplicitamente che il prezzo di acquisto, purtroppo, non può assolutamente essere preso in considerazione, per ovvi motivi cui non accenniamo nemmeno.

Il C/64, ancora in produzione, monta un processore a 8 bit

Prime formule

Possiamo, quindi, tentare una prima valutazione, limitandoci a considerare modelli usati, ma **rigorosamente identici** a quelli acquistabili presso rivenditori autorizzati:

```
Vu = Pn x Cv
```

...in cui **Vu** è il valore dell'apparecchio usato; **Pn** è il prezzo di vendita del modello nuovo; **Cv** è un coefficiente (variabile tra 0 ed 1) che tiene conto della vetustà dell'apparecchio.

La prima perplessità viene proprio dalle considerazioni su **Pn**. E' infatti noto che il prezzo al pubblico di un apparecchio varia a seconda del punto di vendita; altrettanto spesso, poi, l'omaggio di vari programmi e/o accessori (offerti al momento dell'acquisto di un apparecchio nuovo) partecipano a generare confusione.

Purtuttavia bisogna prendere una decisione: possiamo ragionevolmente definire come prezzo di vendita Pn il valore derivante dalla seguente relazione:

```
Pn = Pm x (Pm / Pl)
```

...in cui **Pm** è il prezzo medio di vendita al pubblico (ottenuto, magari, effettuando la media riscontrata presso due coppie di rivenditori, rispettivamente la più cara e la meno cara, in una città di almeno 300 mila abitanti); **Pl** è il prezzo di listino del distributore ufficiale.

Supponiamo, ad esempio, che il prezzo di listino Pl di un computer (**I.V.A. compresa, ovviamente**) sia di 5 milioni; tale modello, però, viene venduto a L. 3.5, 3.7, 4.1, 3.2, 3.6 milioni da altrettanti rivenditori di una grande città (poco importa se alcuni commercianti offrono, in omaggio, software o schede varie).

Il valore medio sarà:

```
Pm =(3.2+ 3.5+ 3.7+ 4.1)/4= 3.625
```

Il valore commerciale del computer nuovo sarà quindi:

```
Pn = 3.625 x (3.625 / 5) = 2.628
```

Il metodo suggerito, come si può notare, penalizza fortemente i listini eccessivamente "gonfiati" rispetto ai prezzi reali

degli apparecchi, malcostume purtroppo diffuso, e non solo in Italia.

Del resto, con il metodo suggerito, ci si tiene al riparo da eccessive "sorprese", soprattutto nel caso di importatori di pochi scrupoli, decisi solo a far soldi.

Nei casi in cui, inoltre, il prezzo medio risulta identico (o quasi) a quello di listino, ci si può fare un'idea sulla serietà dell'importatore (o distributore) che, probabilmente, offre altrettanto seriamente una serie di servizi difficilmente superflui.

Se il metodo suggerito non vi convince, comunque, indicatene un altro, magari più rispondente alle esigenze del mercato e che (visto che ci siete) possa funzionare come deterrente contro importatori improvvisati; in fin dei conti la sfida consiste proprio nel determinare un procedimento che, pur corretto, tenga conto di varie sfaccettature, non ultima la serietà commerciale.

Un vecchio drive 1541: il modello attuale costa la metà

Obsolescenza

Passiamo ora a determinare il coefficiente di vetustà **Cv**.

Quasi ogni anno vengono proposti modelli nuovi che, inevitabilmente, abbattono immediatamente il valore del modello che li ha preceduti.

Ciò è verissimo nel caso di personal computer; un po' meno per accessori molto costosi (scanner, stampanti laser, monitor super professionali) che, tuttavia, risentono ugualmente del passare degli anni.

Il coefficiente Cv, a nostro parere, può variare tra **0.5** e **0.9**; il primo valore (cioè metà del prezzo dello stesso modello oggi venduto) si può applicare a modelli di computer il cui primo esemplare sia stato commercializzato non più tardi di **18 mesi** precedenti alla data in cui si valuta l'apparecchio.

Il valore **0.9**, invece, si può applicare a computer la cui prima commercializzazione risalga a non oltre due mesi precedenti.

Ad esempio, consideriamo un computer che, nell'ottobre 1991, verrà venduto (nuovo) al prezzo di 3 milioni. Il primo esemplare è stato commercializzato (più o meno) a partire da giugno 1990 (anzianità commerciale = 16 mesi); lo stesso computer, usato, può esser valutato non meno di 0.5 x 3 milioni (cioè 1 milione e mezzo) e non più di 0.9 x 3 milioni (2 milioni 700 mila lire) per gli esemplari acquistati nell'estate '91.

La prima cifra si può applicare, senza problemi, ad esemplari acquistati tra giugno e settembre 1990.

Il coefficiente più elevato **dovrebbe** essere applicato ad esemplari acquistati di recente.

Con ogni probabilità, però, entro breve tempo quel particolare modello verrà sostituito da un altro, tecnologicamente più all'avanguardia: quale coefficiente applicare? Alla risposta, ovviamente, dovrete pensarci voi; noi ci siamo limitati a presentare il problema.

Un Amiga 500 prodotto tre anni fa: quanto vale oggi?

Fuori produzione

Il caso esaminato precedentemente, infatti, rivela tutta la sua importanza nella valutazione di modelli usciti di produzione.

Un Amiga 2000, ad esempio, è tuttora in produzione (proprio come il C/64). Sappiamo tutti, però, che i primi esemplari dell'A-2000, a parte le Rom del Kick-Start e la versione del Workbench, montano integrati diversi rispetto ai modelli prodotti di recente.

Anche il "nuovo" C/64, nonostante le funzioni siano rimaste rigorosamente identiche ai primissimi esemplari, è costituito da una scheda nettamente diversa da quella prodotta tempo addietro, se non altro per la notevole customizzazione dei chip portata avanti progressivamente nel tempo.

Inevitabilmente non si può trascurare tale aspetto della questione, soprattutto per quanto riguarda, appunto, la valutazione dell'apparecchio:

L'A-1000 non viene prodotto, ma funziona: come valutarlo?

chi acquista, si sa, tende sempre a tirare sul prezzo, approfittando di argomentazioni che, nel caso specifico, sono più che valide.

Come valutare, quindi, un modello che, pur essendo ancora presente sul mercato del nuovo, risulta, in effetti, diverso da quello attualmente prodotto?

Si potrebbe moltiplicare il valore determinato (per mezzo dei calcoli effettuati fino a questo momento) per un ulteriore coefficiente riduttivo (inferiore, ovviamente, all'unità). Quale debba essere, poi, dovrete scoprirlo voi...

Intrinsecamente

Nel determinare il valore di un apparecchio usato non è possibile trascurare la "quantità" di tecnologia ivi presente.

Un modernissimo (si fa per dire...) computer basato su microprocessore **8088** è di per sé obsoleto e di limitatissimo valore tecnologico.

Tutti sanno, infatti, che presto verremo invasi da computer basati su processori **80486**; questi provvederanno, in breve tempo, ad abbattere i prezzi dei computer basati su **80386** che, a loro volta, costringeranno i produttori di 80286-based a limitare i prezzi; che ne sarà dei computer basati su 8088? Probabilmente non verranno più prodotti.

La stessa sorte toccherà ovviamente, in un prossimo futuro, anche agli elaboratori dotati di 80286 e, un po' più in là, anche ai nuovissimi 80486.

Sarebbe davvero interessante, quindi, stabilire fin da ora una procedura di calcolo che tenga conto anche dell'obsolescenza intrinsecamente posseduta da processori, tuttora in produzione, ma destinati a scomparire dal circuito di massa. Il valore di un computer, quindi, potrebbe essere ritoccato moltiplicandolo per un coefficiente (compreso, ancora, tra 0 ed 1) che tenga conto dell'"attualità" commerciale di un processore.

Oggi come oggi, ad esempio, tale coefficiente potrebbe valere **1** nel caso di computer basati su 80486 e 80386; potrebbe scendere a **0.8** nel caso di 80286-based; per quanto riguarda gli 8088/86 non osiamo pronunciarci: figuratevi che cosa ne pensiamo dei computer basati su **Z-80** e **6502** (e chi ha orecchie per intendere, intenda...).

❑❒

Come partecipare

Stavolta sono invitati a partecipare alla sfida tutti i lettori, anche quelli che il computer non lo posseggono ancora.

L'ideale sarebbe partecipare inviando un programmino interattivo che, mediante domande e risposte, tenga conto di tutti i fattori che, presenti e futuri, possano ragionevolmente influenzare il valore commerciale di un apparecchio usato.

In quest'ultimo caso, naturalmente, l'algoritmo utilizzato non dovrà essere influenzato da particolari esigenze di chi lo scrive: se volete vendere il vostro **C/16** che giace in cantina, non moltiplicate il prezzo di acquisto per un valore superiore all'unità...

Più che altro, tuttavia, ci interessa conoscere il vostro parere, per quanto possibile disinteressato, che faccia capire le considerazioni effettuate da chi deve acquistare o vendere un computer usato. Si potrebbero inserire, è ovvio, correttivi che tengano conto della presenza di eventuali componenti elettromeccanici (posseduti prevalentemente da stampanti, drive, tastiere) maggiormente soggetti ad usura rispetto a schede puramente elettroniche, praticamente "eterne".

La sfida è aperta: a voi la risposta ad un quesito che, apparentemente banale, può rivestire un'importanza non indifferente nel mercato dell'informatica.

Una scheda Super VGA, nel 1989, costava quasi un milione...

di Rodolfo Facchinetti

Chi va piano, va sano e va lontano

Vi spieghiamo in che modo è possibile inserire un "deceleratore", esclusivamente software, per il C/64

Sono ormai finiti i bei tempi in cui nelle edicole abbondavano le cassette colme di "nuovi" giochi (tutti originali?...) per il **C/64**.

Ora vanno di moda **Amiga** ed **Ms - Dos** e per noi 64isti non c'è quasi più niente. Non resta che riesumare le cassette sepolte in qualche scatola e giocare con quelle.

Di tutti i giochi che possediamo, però, non sono molti quelli di cui abbiamo visto la fine. Infatti, trovandoci davanti a difficili passaggi (molto duri da superare), si preferiva lasciar perdere e passare ad altro. Giochi da 200 blocchi (e più) dei quali non siamo mai andati oltre il primo livello, chissà quante magnifiche schermate e quante meravigliose musiche sono in grado di offrirci. Tutto ciò perché, con i nostri riflessi lenti e tardivi, non siamo in grado di competere con il computer. Se, però, nei momenti critici si potesse rallentare l'azione, i nostri occhi avrebbero la possibilità di vedere il pericolo incombente ed intervenire in tempo per evitarlo.

Servirebbe, dunque, un **rallentatore tipo moviola**; ed è appunto ciò che abbiamo intenzione di spiegare in queste righe: come inserire nei vostri giochi (protetti e non) una routine di rallentamento.

(Quasi) senza cartuccia

Per la verità esistono in commercio particolari cartucce in grado di eseguire il lavoro che stiamo per descrivere.

Il problema specifico, comunque, si risolve nel modo più semplice, che è quello di inserire un ciclo di ritardo nel gioco stesso. Dovete sapere, infatti, che i programmatori dei videogames non scrivono un gioco in un'unica routine, ma lo "spezzettano" in tante brevi subroutine: una per muovere gli sprite, un'altra per il punteggio, un'altra ancora per la musica e così via. Una alla volta, poi, vengono eseguite ciclicamente.

In trucco consiste nel far eseguire la routine di rallentamento proprio come se fosse una normalissima routine, simile a tutte le altre.

Passiamo, ora, a spiegare, passo passo, tutto ciò che si deve fare per inserire stabilmente il rallenty in un qualsiasi gioco.

Per giocare, di solito, è necessario inserire un joystick in una delle due porte (1 oppure 2). Nella porta rimasta vuota dovrete inserire un altro joystick, **dotato di autofire**, la cui attivazione darà inizio al rallenty.

Se non possedete un joystick con autofire, incatenate qualche parente alla scrivania e obbligatelo a tener premuto il tasto Fire del secondo joy.

Cominciamo col preparare tutto ciò che servirà, cioè: un block notes, una penna, e necessariamente una **cartuccia sprotettrice** (Niki Cartridge) o simili. Queste cartucce, oltre al copiatore, di solito incorporano anche un monitor per linguaggio macchina che servirà per curiosare nella memoria del computer.

La routine

Ed eccoci, finalmente, a descrivere la routine (riportata nell'apposito riquadro) in cui alcuni dati sono sostituiti da

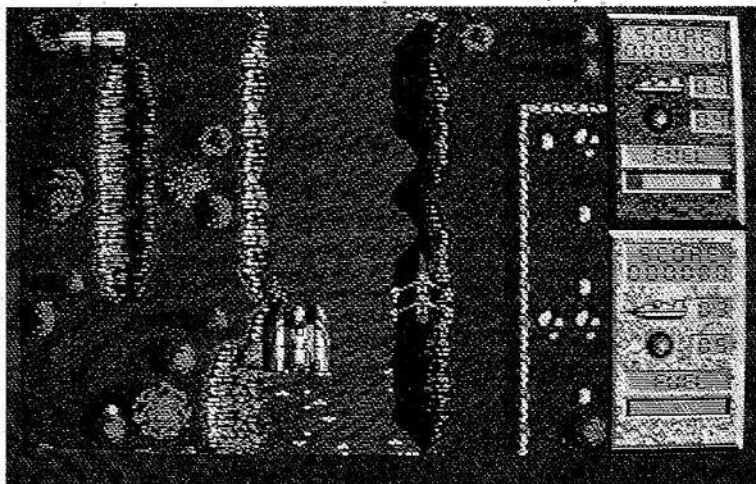
due lineette perché devono essere scelti in base alla struttura del gioco.

Copiate sul blocco note la routine di rallentamento, ma attenzione: **non** dovete copiare le **istruzioni** Assembler, (Lda, Sta, ecc.), ma i **codici esadecimali** che la compongono (**AD – DC**) comprese le lineette. Questo perché con Niki Cartridge (cartuccia che prendiamo come riferimento) il monitor non consente di scrivere nella memoria le istruzioni Assembler, pur se è possibile modificare una ad una le varie locazioni.

Dopo averli trascritti, contateli: dovrebbero esser **23**. A questo punto bisogna rintracciare, nella memoria, una sequenza di byte libera **lunga almeno 23 byte** (oltre a **2** byte per i flag) in cui inseriremo la nostra routine. I due flag (locazioni usate dalla routine come contatori) potremmo anche inserirli in altra parte, ma se sono vicini è meglio.

Inserita la cartuccia, caricate il gioco desiderato, giocate per qualche minuto, quindi "frizzate", cioè congelate il gioco stesso premendo il **pulsante Freeze** presente sulla cartuccia. Dal menu sceglierete il monitor per esaminare l'interno del programma. Dobbiamo cercare, nella memoria Ram, una sequenza di zeri lunga almeno 23 byte.

Quasi tutti i monitor hanno un comando specifico, per cercare una sequenza di byte, che di solito è **H** (da **h**unt, cerca).

Numerosi monitor l.m., disponibili su cartucce (come la Niki) che congelano il gioco, **non** prevedono il comando H. Anche il comando **D** (disassembla) ci è precluso. Così, per forza di cose, per guardare la struttura del programma dobbiamo resettare il computer (resettare, non spegnere!)

Premete dunque il tasto di Reset e sul video apparirà il solito messaggio, come se il computer fosse stato appena acceso; però il programma è ancora presente nella memoria.

Attivate il monitor e digitate...

```
D 0800 9FFF
```

...per far scorrere le istruzioni assembler che compongono il programma. Quando troverete una sequenza di zeri (istruzione **Brk**) abbastanza lunga da contenere la nostra routine (più due byte per i flag) annotatene l'indirizzo (locazione d'inizio) sul blocco note, preferendo il primo spazio libero che trovate dopo $0800. Potrebbero esservi altre zone contenenti zeri, ma libere solo in apparenza (pagina grafica, pagina colore, sprite, note musicali). Inserendo la routine in queste zone, essa funzionerà ugualmente, ma vi accorgerete subito che qualcosa è cambiato (la musica, i colori, ecc.); in questo caso cambiate posto alla routine di rallenty.

Date anche un'occhiata al classico **Buffer del registratore** (per controllare se quest'area non è usata dal programma) nel modo seguente: riempite di zeri il Buffer (da **033C** a **03FB**) e riavviate il gioco. Giocate un po' e poi ricontrollate; se gli zeri sono ancora al loro posto vuol dire che quella è una zona di memoria ottima per inserire la routine di rallenty.

Trovata, in un modo o in un altro, la sequenza di 23 zeri, controllate se ve ne sono ancora due (per un totale di 25) da usare per i flag. Se non c'è posto dovrete sistemarli altrove. Cercate due byte vuoti consecutivi, così non rischierete di metterli in un posto sbagliato.

Quando avete deciso dove metterli, inserite il loro indirizzo nella routine che avete scritto sul blocco note.

Esempio:

```
flag 1 = $38 DE
flag 2 = $38 DF
```

Dividete in:
38 (byte alto)
DE (byte basso)

```
AD -- DC      LDA $DC--      Controlla la pressione del Fire
29 10         AND #$10       alle locazioni DC00 oppure DC01.
D0 0F         BNE a RTS      Se non è premuto torna al gioco.
CE -- --    * DEC flag 1     Rallentamento: cioè si
D0 FB         BNE a *        decrementa il flag 1. Giunti a zero
CE -- --      DEC flag 2     si prosegue decrementando il flag 2
D0 F6         BNE a *        e poi ricomincia. Azzerato il flag 2
A9 40         LDA #$40       lo si ricarica col valore $40 quindi
8D -- --      STA flag 2     e si torna al gioco. Ricaricando il
60            RTS            flag 2 con un valore maggiore di $40
                             si avrà un rallenty più marcato.
```

La semplice subroutine l.m. per rallentare un gioco
Per la digitazione dei codici è indispensabile leggere con la massima attenzione le informazioni descritte in queste pagine

La quarta istruzione della nostra routine...

```
DEC flag 1
```

...diverrà...

```
CE DE 38
```

...e la sesta

```
CE DF 38
```

In ogni caso trascrivere **prima** il byte basso, **poi** quello alto.

Poiché è molto difficile trovare la routine di comando, cioè quella che si occupa di richiamare ciclicamente le subroutine che formano il programma principale, dovremo cercare una delle subroutine, chiamate dal gioco stesso, che presenti, dopo il suo termine, **due spazi "vuoti"** nei quali inserire la chiamata alla nostra routine.

Digitiamo, dunque:

```
H 0800 BFFF 60 00 00
```

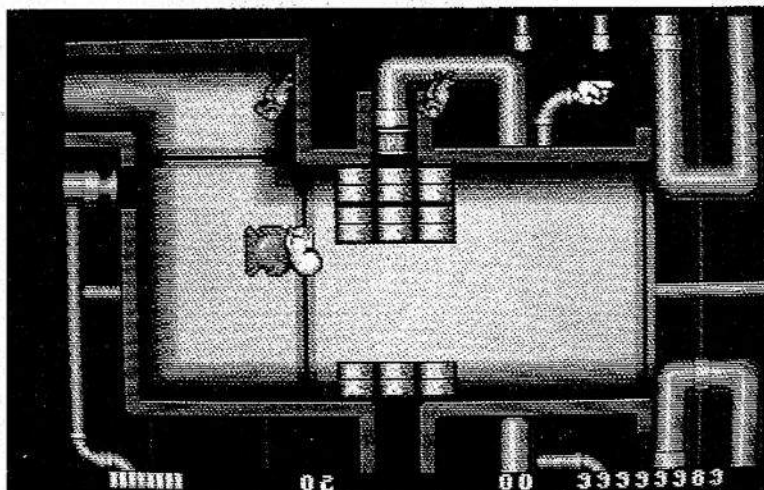
...istruzione che ricerca, tra $0800 e $BFFF, un'istruzione **Rts** seguita da due caselle vuote; oppure...

```
H 0800 BFFF EA EA EA
```

...che cerca, sempre tra $0800 e $BFFF, tre istruzioni **Nop** consecutive (ricordiamo che Nop = **n**essuna **op**erazione) cioè tre byte da utilizzare per la chiamata, che sarà, in questo caso, leggermente diversa. I tre Nop dovranno essere necessariamente all'interno di una subroutine.

Eseguito il comando **H** appariranno, sullo schermo, le locazioni d'inizio di tali sequenze. Se non ne compare nessuna, tenteremo in un altro modo (come vedremo più avanti).

Annotate anche questi indirizzi e verificate che siano idonei al nostro scopo. E' necessario che Rts faccia veramente parte di una subroutine e non sia un valore numerico (**$60**) qualunque della pagina grafica o una nota musicale. Controllate la routine a cui appartiene, disassemblando a partire da alcune istruzioni presenti prima degli indirizzi che avete annotato in precedenza. Dovete trovarvi in presenza di una sequenza di istruzioni assembler priva di punti interrogativi (dati senza significato); controllate anche che non vi siano istruzioni `Sta -- --` dirette alle due locazioni dopo l'Rts; ciò significherebbe che le suddette locazioni sono usate dalla routine, magari come indirizzo indiretto (per i tre Nop, di solito, non dovrebbe essrci alcun problema).

Finalmente trovato il posto giusto, annotatelo, ricaricate il gioco, giocate un po', quindi frizzate e scegliete il monitor da menu. A questo punto inserite la routine nello spazio che avete prima individuato.

Usando il comando **M** (Memoria) digitate:

```
M (indirizzo routine)
```

Appariranno 8 byte che modificherete in **AD 00 DC** oppure **AD 01 DC** a seconda se il joystick non usato per giocare sia in porta 2 oppure in porta 1. Per scrivere i dati, basta posizionarsi sui byte da cambiare, scrivere i nuovi dati e premere Return. Continuiamo con **29 10** e **D0 0F** poi **CE flag 1**.

Una volta modificati 8 byte, premete Return due volte ed il cursore in basso; vedrete altri 8 byte vuoti.

Proseguiamo con D0 FB - CE flag 2 - D0 F6 - A9 40 - 8D flag 2 - 60 ed è finita.

Controllate ciò che avete scritto e tirate un gran respiro. Ancora un ultimo sforzo. Bisogna inserire la chiamata alla nostra routine. Visualizziamo uno degli indirizzi annotati.

Abbiamo **60 00 00** oppure **EA EA EA**.

Nel primo caso trasformiamo i tre dati in **Jmp** - indirizzo rallenty, cioè: **4C -- --** (prima il byte basso!)

Nel secondo caso (EA EA EA) trasformiamo i tre dati in JSR - indirizzo rallenty, cioè: **20 -- --** (prima, sempre, il byte basso).

Fatto ciò, fate ripartire il gioco e auguri. Se non dovesse funzionare al primo colpo, cambiate il posto alla chiamata senza dimenticare di rimettere a posto i byte che avete modificato.

Se non avete trovato un posto per la chiamata nei modi descritti, potete cercare all'interno di una subroutine un salto (**Jsr -- --**) verso una "qualsiasi destinazione". Inserite al posto di questa istruzione, un salto alla routine di rallenty (Jsr - rallenty) cioè **20 -- --** (ricordarsi di anteporre il byte basso).

Poi inserite nella routine di rallenty (prima dell'Rts) l'istruzione che avete cambiato (Jsr - "qualsiasi destinazione") cioè **20 -- --**

Pertanto la nostra routine si allungherà di tre byte e terminerà con **JSR -- -- RTS (20 -- -- 60)**

Conclusioni

La tecnica descritta in queste pagine, come si può immaginare, non può essere considerata "universale", ma solo un suggerimento sul modo di agire per inserire una nostra (possibilmente breve) rutine all'interno di programmi lunghi e complessi, come può essere, appunto, un videogame.

Il difficile, ovviamente, è rappresentato dall'individuare una routine che venga spesso (magari sempre) richiamata durante l'esecuzione del programma; il pensiero va immediatamente alle routine inserite nel **ciclo di Interrupt**, facendo però attenzione a non esagerare nell'impostare la lunghezza dell'"intrusa" per evitare che il computer si limiti ad elaborare eslusivamente... le routine contenute in quest'ultimo ciclo!

di Giancarlo Mariani

Tutti i codici dei vostri file

Tre programmi, in altrettanti linguaggi, consentono di esaminare il contenuto di un qualsiasi file memorizzato su dichetto

Usando **Ms - Dos**, sicuramente tutti prima o poi si saranno imbattuti nel comando **TYPE** che serve per visualizzare il contenuto di un file, purchè in formato ASCII.

Viene qui proposta una procedura (in **C**, in **Pascal** ed in **Basic**) che permette di ottenere un Type un po' particolare dal momento che non si limita a visualizzare il file specificato in formato ASCII, ma lo evidenzia anche in **esadecimale**, consentendo così di esaminare alcuni caratteri non-ascii che altrimenti non potrebbero essere visualizzati. Solitamente questo tipo di visualizzazione si chiama **Dump** del file.

Le tre routine consentono sia di dare una ripassata alla gestione dei files nei tre linguaggi, sia di meglio comprendere come **passare parametri** da Dos ai programmi utente.

L'utilizzo della procedura è molto semplice: una volta digitato il programma, salvato con il nome **Mostra** e compilato (prodotta, cioè, la versione EXE), da Dos si deve semplicemente digitare...

`Mostra Nomefile`

...dove **`NomeFile`** è il nome del file che vogliamo visualizzare. A questo punto, ammesso che il file esista, lo vedremo DUMPato sullo schermo (o sulla stampante se si digita `Mostra nomefile > PRN`).

Prendiamo, come esempio, un "probabile" file `AUTOEXEC.BAT`, certamente presente anche sul vostro computer (pur se con le dovute personalizzazioni). Eseguendo da Dos il comando `Type Autoexec.bat`, il risultato sarebbe come in **figura 1**.

Usando, invece, uno dei tre programmi presentati, l'output sarà simile a quello presentato in **figura 2**.

Il file viene, cioè, visualizzato in **blocchi** (o record) di **128** bytes alla volta, con la codifica esadecimale e ASCII del suo contenuto. Se il carattere non è interpre-

```
TYPE C:\AUTOEXEC.BAT

;d:\dos;d:\qb4;d:\tc\bin
fast
dosedit
set LIB=d:\qb4
set TEMP=c:\windows\temp
set NC=d:\tools
pc-cache /SIZEXT=128
```

Figura 1
Esempio di contenuto del file Autoexec.bat

```
MOSTRA C:\AUTOEXEC.BAT

Record : 0001
         0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00000000 40 65 63 68 6F 20 6F 66 66 0D 0A 70 61 74 68 20 @echo off..path
00000010 63 3A 5C 3B 64 3A 5C 64 6F 73 3B 64 3A 5C 62 61  c:\;d:\dos;d:\ba
00000020 74 3B 64 3A 5C 74 6F 6F 6C 73 3B 64 3A 5C 71 62 t;d:\tools;d:\qb
00000030 34 3B 64 3A 5C 74 63 5C 62 69 6E 3B 64 3A 5C 74  4;d:\tc\bin;d:\t
00000040 70 5C 62 69 6E 3B 63 3A 5C 77 69 6E 64 6F 77 73   p\bin;c:\widows
00000050 3B 64 3A 5C 6D 61 73 6D 0D 0A 70 72 6F 6D 70 74  ;d:\masm..prompt
00000060 20 24 70 24 67 0D 0A 66 61 73 74 0D 0A 64 6F 73   $p$g..fast..dos
00000070 65 64 69 74 0D 0A 73 65 74 20 4C 49 42 3D 64 3A   edit..setLIB=d:


Record : 0002
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00000080 5C 71 62 34 0D 0A 73 65 74 20 54 45 4D 50 3D 63   \qb4..setTEMP=c
00000090 3A 5C 77 69 6E 64 6F 77 73 5C 74 65 6D 70 0D 0A   :\widows\temp..
000000A0 73 65 74 20 4E 43 3D 64 3A 5C 74 6F 6F 6C 73 0D   set NC=d:\tols.
000000B0 0A 70 63 2D 63 61 63 68 65 20 2F 53 49 5A 45 58   .pc-cache /SZEX
000000C0 54 3D 31 32 38 0D 0A 00 00 00 00 00 00 00 00 00  T=128..........
000000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figura 2
Visualizzazione possibile con uno dei tre programmi pubblicati

```
' MOSTRA.BAS : Mostra un file in formato HEX/ASCII (QBasic 4.0)
' Uso: MOSTRA drive:path\nomefile.est
'
DEFINT A-Z
DECLARE SUB MostraRecord (Record AS INTEGER, Punta AS LONG)
DECLARE SUB Mostra16 (InizioByte AS LONG, Indice AS INTEGER)

DIM SHARED LenRec AS INTEGER              'Lunghezza record
LenRec = 128

DIM File AS STRING                        'Nome del file
DIM Stato AS INTEGER                      'Stato in lettura file
DIM Record AS INTEGER                     'Record del file
DIM Punta AS LONG                         'Puntatore al file
DIM SHARED Buffer(1 TO LenRec) AS STRING * 1
                                          'Buffer per un record
DIM k AS INTEGER                          'Variabile per ciclo FOR
DIM LunghezzaFile  AS LONG                'Lunghezza del file

Record = 1                                'Inizializzazioni
Punta = 0
' Controlla il parametro passato al programma
File = LTRIM$(RTRIM$(COMMAND$))
IF File = "" THEN
  PRINT "MOSTRA: Specificare il nome del file"
  END
END IF

' Apre il file specificato, controllando gli errori
Erv = 0
ON ERROR GOTO Errore
OPEN File FOR INPUT AS #1
ON ERROR GOTO 0
CLOSE #1
IF Erv <> 0 THEN
  PRINT "MOSTRA: Errore nell'apertura del file "; File
  END
END IF

PRINT "MOSTRA file "; File
OPEN File FOR BINARY AS #1
LunghezzaFile = LOF(1)

AltroRecord:

  ' Carica dal file un record lungo LenRec
  ' e lo mette nella variabile Buffer()
  Stato = 0
  FOR k = 1 TO LenRec
    IF (k + Punta <= LunghezzaFile) THEN
      GET #1, k + Punta, Buffer(k)
    ELSE
      Stato = 1: Buffer(k) = CHR$(0)
    END IF
  NEXT k

  ' Se il file non e' finito, mostra il record
  CALL MostraRecord(Record, Punta)
  Punta = Punta + LenRec: Record = Record + 1
  IF Stato = 0 THEN GOTO AltroRecord

PRINT : PRINT
CLOSE #1
```

tabile in formato ASCII, viene visualizzato, al suo posto, un punto. Vediamo ora di analizzare i 3 listati proposti.

Mostra.bas

Programma principale:

A) Le prime righe (`Declare Sub...`) servono per dichiarare le subroutines e le funzioni usate dal programma. Questo permette, al compilatore, di controllare il corretto "passaggio" di parametri tra le varie subroutines ed il programma principale.

B) Le successive `Dim` servono per dichiarare le variabili usate. Anche se questa parte di dichiarazione, in Basic, non è obbligatoria, è sempre opportuno farla allo scopo di aver chiaro il numero, il nome ed il tipo di tutte le variabili usate. Il parametro `Shared` significa che le variabili sono dichiarate come globali, ossia possono essere viste sia dal programma principale che da tutte le subroutines o funzioni.

C) Le successive righe prendono il parametro passato al programma da Dos (ricordiamo che la sintassi è `Mostra NomeFile`). Per fare ciò viene utilizzata una variabile di sistema del Basic, ossia la variabile **Command$**. Questa variabile, all'atto del richiamo del programma, contiene il (o gli) eventuali parametri passati a quest'ultimo. Tramite Command$, pertanto, sarà possibile conoscere il nome del file "passato" al programma.

D) Nella successiva fase il programma controlla che il file da visualizzare esista. Per fare ciò, questo viene aperto in input, controllando l'eventuale errore che ne deriva.

Dal momento che un'apertura, in input, di un file che non esiste produrrebbe un run-time error (ed il programma si bloccherebbe), viene usata l'istruzione `On Error Goto` per far saltare il programma alla routine specificata in caso di errore.

La routine (chiamata, nel nostro caso, `Errore`) provvede ad inserire, in una variabile, il numero corrispondente all'errore occorso. Il numero è preso dalla variabile di sistema **Err**. L'istruzione `Resume Next` fa riprendere quindi il programma dall'istruzione immediatamente successiva a quella che ha generato l'errore.

`On Error Goto 0` riabilita, quindi, il normale controllo degli errori del Basic.

```
END

END
Errore:
  Erv = ERR
RESUME NEXT
' Mostra 16 bytes del file in HEX ed in ASCII
'     InizioByte : offset attuale nel file
'     Indice     : Puntatore all'interno del buffer del file
SUB Mostra16 (InizioByte AS LONG, Indice AS INTEGER)

DIM j AS INTEGER                    'Offset
DIM c AS STRING * 1                 'Carattere da scrivere

PRINT RIGHT$("00000000" + HEX$(InizioByte - 1), 8); "    ";
' Offset corrente

' Mostra 16 bytes del file in esadecimale
FOR j = 0 TO 15
  PRINT " "; RIGHT$("00" + HEX$(ASC(Buffer(Indice + j))), 2);
NEXT j: PRINT "    ";

' Mostra 16 bytes del file in ASCII
FOR j = 0 TO 15
  c = Buffer(Indice + j)
  IF c < CHR$(32) OR c > CHR$(126) THEN c = "."
  PRINT c;
NEXT j
PRINT

END SUB

' Mostra un record del file.
'    Record : Numero del record
'    Punta  : Puntatore al file
'    LenRec : Lunghezza del record
SUB MostraRecord (Record AS INTEGER, Punta AS LONG)

DIM Indice AS INTEGER
PRINT : PRINT "Record : "; RIGHT$("0000" + HEX$(Record), 4)
PRINT "               0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F"

FOR Indice = 1 TO LenRec STEP 16
  CALL Mostra16(Punta + Indice, Indice)
NEXT Indice

END SUB
```

Fine del listato Qbasic

```
{ MOSTRA.PAS : Mostra su video un file }
{in formato HEX/ASCII (TurboPascal 5.5)}
{ Uso: MOSTRA drive:path\nomefile.est }

function Hex(Number:longint; Digit:integer):string; forward;
procedure MostraRecord (Rec:integer; Punta:longint); forward;
procedure Mostra16 (InizioByte:longint; Indice:integer); forward;
const LenRec=128;                        { Lunghezza record }
var
```

E) Una volta controllato che il file esiste, questo viene chiuso e quindi riaperto, questa volta in binario. Tale tipo di apertura consente di analizzare non solo files in formato ASCII ma anche files .COM, .EXE, o di qualsiasi altro tipo. La funzione `LOF (FileNumber)` restituisce la lunghezza, in bytes, del file specificato.

F) Il ciclo `For... Next` (da 1 alla lunghezza del record specificato, nel nostro caso 128 bytes) carica effettivamente dal file i caratteri e li pone nell'array **Buffer()**. Nel caso in cui sia stata raggiunta la fine del file (controllata tramite la lunghezza), il buffer viene riempito di **Chr$(0)**. L'istruzione per caricare un byte dal file aperto in modo binario è...

`GET #FileNumber, Puntatore, Variabile`

...dove **FileNumber** è il numero del file dal quale vogliamo caricare, **Punta** è il puntatore (il numero del byte) al byte da caricare, mentre **Variabile** è una variabile di tipo stringa, lunga 1 carattere, che conterrà il byte caricato.

Nel caso in cui il file sia giunto alla fine, la variabile **Stato** conterrà 1.

G) Successivamente viene richiamata la procedura `MostraRecord`, che mostra il record del file contenuto nell'array Buffer, quindi aggiunge, al puntatore, il numero di bytes componenti il record ed incrementa il numero del record. Il ciclo a questo punto ricomincia.

MostraRecord

Questa subroutine serve a visualizzare il record contenuto nella variabile Buffer.

A) `Dim Indice`, presente nella subroutine, dimensiona la variabile **Indice** all'interno della stessa. Questa è una variabile che sarà vista solo da questa subroutine, e quindi non da altre subroutines o dal programma principale.

B) Le successive **Print** preparano la maschera per la visualizzazione. `Print Right$...` serve per visualizzare un numero lungo sempre 4 cifre. Infatti, aggiungendo in "testa" al numero quattro zeri e considerando le sole quattro cifre presenti a destra, saremo sicuri che il numero visualizzato sarà sempre lungo 4 cifre. La funzione `Hex$` converte in esadecimale il numero specificato.

C) Il successivo ciclo For serve per visualizzare una riga lunga 16 bytes alla volta, visualizzazione effettuata tramite la routine Mostra16.

```
  FileName:string;                    { Nome del file }
  Stato:integer;                      { Stato in lettura file }
  Rec:integer;                        { Record del file }
  Punta:longint;                      { Puntatore al file }
  Buffer:array [1..LenRec] of byte;   { Buffer per un record }
  k:integer;                         { Variabile per ciclo FOR }
  LunghezzaFile:longint;                 { Lunghezza del file }
  Fil:file of byte;                      { Variabile per il file }
label AltroRecord;

{ Converte un long integer in esadecimale }
{   Number : numero da convertire        }
{   Digit  : lunghezza del risultato     }
function Hex(Number:longint; Digit:integer):string;
const
  ArrHex='0123456789ABCDEF';
var
  Temp:string;
  TH,TL:integer;
  B0,B1,B2,B3,Nh,Nl:byte;
begin
 Temp:='';
 TH:=Number div 65536;
 TL:=Number-(TH*(65536));
 B0:=Lo(TL); B1:=Hi(TL); B2:=Lo(TH); B3:=Hi(TH);
 Nh:=B3 div 16; Nl:=B3-(Nh*16);
 Temp:=Temp+copy(ArrHex,Nh+1,1); Temp:=Temp+copy(ArrHex,Nl+1,1);
 Nh:=B2 div 16; Nl:=B2-(Nh*16);
 Temp:=Temp+copy(ArrHex,Nh+1,1); Temp:=Temp+copy(ArrHex,Nl+1,1);
 Nh:=B1 div 16; Nl:=B1-(Nh*16);
 Temp:=Temp+copy(ArrHex,Nh+1,1); Temp:=Temp+copy(ArrHex,Nl+1,1);
 Nh:=B0 div 16; Nl:=B0-(Nh*16);
 Temp:=Temp+copy(ArrHex,Nh+1,1); Temp:=Temp+copy(ArrHex,Nl+1,1);
 Hex:=copy(Temp,9-Digit,Digit);
end;
{ Mostra un record del file.        }
{   Record : Numero del record      }
{   Punta  : Puntatore al file      }
{   LenRec : Lunghezza del record   }
procedure MostraRecord (Rec:integer; Punta:longint);
var
  Indice:integer;
  RecL:longint;
begin
  writeln;
  RecL:=Rec;
  writeln('Record : ',Hex(RecL,4));
  writeln('             0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F');

  Indice:=1;
  repeat
    Mostra16(Punta+Indice,Indice);
    Indice:=Indice+16;
  until Indice>LenRec;
end; { MostraRecord }

{ Mostra 16 bytes del file in HEX ed in ASCII                        }
{    InizioByte : offset attuale nel file                            }
{    Indice     : Puntatore all'interno del buffer del file          }
{                                                                    }
procedure Mostra16 (InizioByte:longint; Indice:integer);
var
  j,i:integer;                  { Offset }
```

Mostra16

Questa routine visualizza 16 bytes sia in esadecimale che in ASCII.

A) Le righe Dim servono, come visto prima, a dimensionare delle variabili da usare solo all'interno della subroutine.

B) La successiva Print scrive l'offset relativo al file, in formato di numero esadecimale lungo 8 caratteri.

C) Il primo ciclo For (da 0 a 15) visualizza 16 bytes componenti la riga in formato esadecimale (2 caratteri).

D) Il secondo ciclo For, sempre da 0 a 15, visualizza gli stessi 16 bytes in formato ASCII. Se il carattere va fuori dai limiti di visualizzazione (**32 - 126**) verrà visualizzato un punto. Questo per evitare la visualizzazione di caratteri strani, tipo Chr$(7), Chr$(13), Chr$(10) ed altri che produrrebbero su video scompensi in visualizzazione.

Mostra.PAS

Programma principale

A) Anche nel caso del **programma Pascal** è opportuno dichiarare le procedure e le funzioni usate all'interno del programma, scrivendo l'intestazione della procedura o funzione seguita dalla parola chiave `forward`.

B) Le successive righe servono, anche in questo caso, per la dichiarazione delle costanti e delle variabili usate dal programma. Come si può notare, la dichiarazione delle variabili nei vari linguaggi è molto simile.

C) Nel caso del Pascal, per passare un parametro da Dos al programma si usa la variabile di sistema **ParamStr**. In questo caso il primo elemento della variabile (`ParamStr(1)`) conterrà il nome del file da mostrare su video. Viene effettuato quindi un controllo sull'effettivo contenuto del parametro.

D) Il successivo passo è un controllo dell'esistenza del file specificato, in modo simile a quello utilizzato in Basic. Il file viene infatti aperto in lettura, e quindi viene controllato l'eventuale errore occorso. Le direttive **{$I-}** e **{$I+}** servono, rispettivamente, per disabilitare e per riabilitare il **controllo dell'errore** da parte del compilatore Pascal. Omettendo queste righe, e cercando di aprire un file che

```
  c:char;                        { Carattere da scrivere }
begin
 write(Hex(InizioByte - 1, 8)); write('   '); { Offset corrente }

{ Mostra 16 bytes del file in esadecimale }

  for j := 0 to 15 do
    write (' ',Hex(Buffer[Indice + j],2));
  write('   ');

{ Mostra 16 bytes del file in ASCII }

  for j := 0 to 15 do
  begin
    i := Buffer[Indice + j];
    if (i < 32) or (i > 126) then i:= 46;
    write(chr(i));
  end;
  writeln;
end; { Procedure Mostra16 }

begin { Main Program }
  Rec := 1;                              { Inizializzazioni }
  Punta := 0;

{ Controlla il parametro passato al programma }

  FileName := ParamStr(1);
  if FileName = '' then
  begin
    writeln('MOSTRA: Specificare il nome del file');
    halt;
  end;

{ Apre il file specificato, controllando gli errori }
{ $I- disattiva errori di I/O, $I+ riattiva }

  {$I-}
  assign(Fil,FileName);
  reset(Fil);
  close(Fil);
  {$I+}
  if (IOresult<>0) then
  begin
    writeln('MOSTRA: Errore nell"apertura del file ',FileName);
    halt;
  end;

  writeln('MOSTRA file ', FileName);
  assign(Fil,FileName);
  reset(Fil);
  LunghezzaFile := FileSize(Fil);

AltroRecord:

  { Carica dal file un record lungo LenRec e lo mette}
   { nella variabile Buffer[] }

  Stato := 0;
  for k := 1 to LenRec do
    if (k + Punta <= LunghezzaFile) then
    begin
      read (Fil,Buffer[k]);
```

non esiste, il compilatore produrrebbe run-time error.

Usando le direttive indicate, dovremo invece controllare noi da programma l'eventuale errore, utilizzando la variabile di sistema **IOresult** che, se diverso da zero, specifica che si è verificato un errore di I/O.

E) Una volta controllata l'esistenza del file, questo viene chiuso e quindi riaperto per permetterne la lettura. La funzione `FileSize`, al pari della LOF del Basic, restituisce la lunghezza del file specificato.

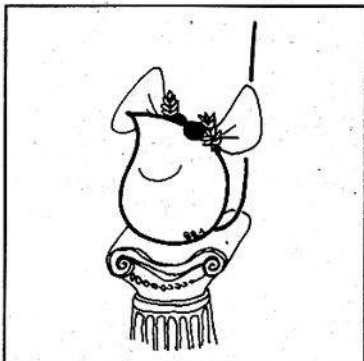
F) Il successivo ciclo For è esattamente la copia di quello del Basic, ossia carica un byte dal file e lo inserisce nella variabile **Buffer[]** fintantochè non si è superata la lunghezza del file, altrimenti nella variabile Buffer[] viene inserito **0**. L'istruzione per leggere un byte dal file è...

```
read (Fil, Variabile);
```

...dove **Fil** è il file dal quale vogliamo leggere, mentre **Variabile** è la variabile intera contenente il byte letto dal file. A differenza del Basic, qui **non** è necessario specificare il puntatore, perchè questo viene incrementato automaticamente ad ogni istruzione **read**. In caso di fine del file, la variabile **Stato** viene posta a 1.

G) Le successive righe richiamano la procedura `MostraRecord`, che serve per mostrare un record del file, quindi il puntatore viene incrementato (in questo caso il puntatore serve solo in visualizzazione) e quindi viene incrementato anche il numero di record, ed il ciclo ricomincia.

Hex

Questa **funzione** serve per convertire un numero da **long integer** a **esadecimale**, ed è l'equivalente della Hex$ del

```
    end
    else begin
      Stato := 1;
      Buffer[k] := 0;
    end;

  { Se il file non e' finito, mostra il record }
  MostraRecord(Rec, Punta);
  Punta := Punta + LenRec;
  Rec := Rec + 1;
  if Stato = 0 then goto AltroRecord;

  writeln; writeln;
  close(Fil);
end.
```

Fine del listato Pascal

```
/* MOSTRA.C : Mostra su video un file */
/*in formato HEX/ASCII (TurboC 2.0)   */
/* Uso: MOSTRA drive:path\nomefile.est*/

#include <stdio.h>
#include <process.h>
#include <string.h>

void MostraRecord (char *Buffer,int Rec,long Punta);
void Mostra16 (long InizioByte,unsigned char *Pointer);

#define LenRec 128                        /* Lunghezza record */

/*                                     */
/* Mostra un record del file.          */
/*   Record : Numero del record        */
/*   Punta  : Puntatore al file        */
/*   LenRec : Lunghezza del record     */
/*                                     */
void MostraRecord (char *Buffer,int Rec,long Punta)
{
  int Indice;

  printf("\n\nRecord : %04X",Rec);
  printf("\n               0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F");

  for (Indice=0; Indice<LenRec; Indice=Indice+16)
    Mostra16(Punta+Indice,Buffer+Indice);
}

/* Mostra 16 bytes del file in HEX ed in ASCII                     */
/*    InizioByte : offset attuale nel file                         */
/*    PuntBuf    : Puntatore all'interno del buffer del file       */

void Mostra16 (long InizioByte, unsigned char *PuntBuf)
{
  int j;                    /* Offset */
  char c;                   /* Carattere da scrivere */

  printf("\n%08lX   ",InizioByte); /* Offset corrente */
```

Basic. A differenza di una procedura, la funzione restituisce un valore, ed in questo caso restituisce il valore esadecimale convertito. La procedura per la conversione non è la più breve, ma è veloce. Il numero viene scomposto negli 8 digit che lo compongono, nel formato basso / alto e quindi, tramite un array, vengono scelte le cifre esadecimali corrispondenti al digit prescelto. In questa procedura sono da segnalare tre istruzioni:

HI, che restituisce la parte alta del numero intero specificato, e **LO**, che restituisce la parte bassa.

Copy è equivalente alla **Mid$** del Basic, e restituisce una sottostringa della stringa specificata. Bisogna quindi specificare il numero di digit (cifre) che si desidera ottenere in uscita.

MostraRecord

La procedura serve per visualizzare un record di 128 bytes estraendoli dal file specificato.

A) La dichiarazione di variabili (`var....`) serve per specificare le variabili da usare solo all'interno della procedura, quindi non visibili da altre procedure o dal main program.

B) Le successive `writeln` servono per preparare la maschera di visualizzazione. E' da notare che per scrivere il numero del record viene richiamata la funzione Hex vista prima.

C) A questo punto viene sviluppato un ciclo `repeat... until` da 1 alla lunghezza del record, a "passi" di 16 bytes alla volta; subito dopo viene richiamata la procedura Mostra16 che visualizza, a mano a mano, 16 bytes del file.

Mostra16

Questa procedura serve per visualizzare 16 bytes del file in formato esadecimale ed ASCII.

A) Anche in questo caso vengono dichiarate alcune variabili da usare solo all'interno della procedura.

B) Il primo `write` scrive l'offset corrispondente ai bytes visualizzati (in formato esadecimale ad 8 cifre).

C) Il primo ciclo For (da 0 a 15) serve per visualizzare i 16 bytes in formato esadecimale (2 cifre).

D) Il secondo For, sempre da 0 a 15, visualizza gli stessi 16 bytes in formato ASCII. Se il carattere va fuori dai limiti per una corretta visualizzaizone (32 - 126) viene sostituito da un punto, per evitare scompensi in visualizzazione.

```
/* Mostra 16 bytes del file in esadecimale */

  for (j=0; j<16; j++)
    printf(" %02X",PuntBuf[j]);
  printf("    ");

/* Mostra 16 bytes del file in ASCII */

  for (j=0; j<16; j++)
  {
    c=PuntBuf[j];
    if ((c<32) || (c>126)) c='.';
    putchar(c);
  }

} /* Mostra16 */

void main(int argc,char *argv[])    /* Main Program */
{
  int Stato=0;                      /* Stato in lettura file */
  int Rec=0;                        /* Record del file */
  long Punta=0;                     /* Puntatore al file */
  char Buffer[LenRec];              /* Buffer per un record */
  FILE *Fil;                        /*  Variabile per il file */

/* Controlla il parametro passato al programma */

  if (strlen(argv[1])==0)
  {
    printf("MOSTRA: Specificare il nome del file\n");
    exit(1);
  }

/* Apre il file specificato, controllando gli errori */

  if ((Fil=fopen(argv[1],"rb"))==NULL)
  {
    printf("MOSTRA: Errore nell'apertura del file %s\n",argv[1]);
    exit(1);
  }

  printf("MOSTRA file %s\n", argv[1]);

  /* Carica dal file un record lungo LenRec */
  /* e lo mette nella variabile Buffer[]    */

  while ((Stato=fread(Buffer,1,LenRec,Fil))!=0)
  {
    MostraRecord(Buffer,++Rec,Punta);
    Punta=Punta+LenRec;
  }

  if (Stato==0) Stato=0;          /* Elimina warning */
  if (argc==0) argc=0;            /* Elimina warning */
  printf("\n\n");
  fclose(Fil);
}
```

Fine del listato in "C"

Mostra.C

Infine passiamo alla descrizione del listato in linguaggio C, leggermente diverso dagli altri, dal momento che il C consente, a volte, una scrittura più "abbreviata" di istruzioni e procedure.

Programma principale

A) Le righe `#include...` servono per includere, nel programma, le varie dichiarazioni delle istruzioni standard del C, utili al compilatore per controllare la corretta digitazione dei comandi C.

B) Anche in C è opportuno dichiarare le procedure e funzioni usate all'interno del programma, scrivendo semplicemente l'intestazione della procedura o funzione seguita da un punto e virgola (vedere le due righe `void....`)

C) La riga `#define...` è equivalente alla **const** del Pascal, ed in questo caso definisce una costante che può essere vista sia dal programma principale che da tutte le altre subroutines e funzioni presenti. La costante definita è **LenRec**, ossia lunghezza del record, posta a 128 bytes.

D) Il programma principale inizia con la parola chiave **main**, eventualmente preceduta dalla parola **void**, che specifica che il main non restituisce al Dos alcun parametro. I due parametri **argc** e **argv** specificati nella dichiarazione del main precisano che il Dos può passare alcuni parametri al programma C, e che questi verranno inseriti nelle variabili specificate. In particolare, argc conterrà **il numero** di parametri passati al programma, mentre **argv[]** è un array di stringhe, ognuna contenente un parametro passato dal Dos.

E) Le successive righe hanno un duplice scopo: anzitutto, servono per **dichiarare** le variabili usate dal programma (dato che le dichiarazioni sono all'interno del main, verranno viste solo dal main); il secondo scopo è quello di **inizializzare** le variabili. Il C consente l'inizializzazione delle variabili direttamente nella dichiarazione, risparmiando così righe di listato sorgente.

Tutte le variabili usate nei programmi C vanno inizializzate perchè il compilatore C, a differenza di altri compilatori, non provvede a compiere automaticamente l'operazione; una variabile, se non dichiarata, potrebbe contenere valori casuali se non inizializzata correttamente.

F) A questo punto viene effettuato il controllo sul parametro passato dal Dos (che è poi il nome del file da mostrare) contenuto nella variabile stringa **argv[1]**. **Argv[0]** contiene **sempre**, di default, il drive e la directory (quindi, il **path**) ove risiede il programma chiamato.

L'istruzione `exit` fa terminare immediatamente il programma (restituendo al Dos il codice specificato tra parentesi) che può essere controllato da Dos tramite eventuali istruzioni batch (If Errorlevel...) per capire se il programma è andato a buon fine oppure no.

G) Per aprire il file viene utilizzata l'istruzione `fopen` che restituisce il puntatore al file, oppure NULL nel caso che il file non esista. Come si vede dal listato, il puntatore viene assegnato alla variabile **Fil** direttamente nell'istruzione If. Questa è una scrittura abbreviata consentita nel C. L'equivalente sarebbe costituita dalla serie di istruzioni...

```
Fil = fopen (argv[1], "rb");
if (Fil==NULL)..........
```

Il parametro **"rb"** in fopen specifica che il file deve essere aperto in lettura (r) ed in modo binario (b).

H) Se il file esiste, tramite l'istruzione `Fread` viene caricato un record (128 bytes) alla volta. L'istruzione fread si usa in questo modo:

```
N = fread (Buffer, Nelem,
     Lunghezza, File);
```

...in cui **Buffer** è la variabile nella quale andranno a finire i caratteri letti dal file. **Lunghezza** specifica quanti caratteri è lungo l'elemento da caricare. **Nelem** specifica quanti elementi sono da caricare. Quindi il numero totale di bytes, letti da fread, risulta Nelem x Lunghezza. File è il file dal quale caricare i bytes.

Fread restituisce un numero intero (N) che specifica quanti elementi (e **non** quanti bytes!) sono stati caricati. Quindi, se fread restituisce 0, vuol dire che è stata raggiunta la fine del file.

Anche in questo caso è stata usata una forma di scrittura abbreviata consentita dal C, dove il numero di elementi letti viene assegnato alla variabile **Stato** direttamente nell'istruzione **While**. La serie di istruzioni corrispondente, non usando la forma abbreviata, sarebbe...

```
Stato = fread (Buffer, 1,
     LenRec, Fil);
while (Stato!=0)
{
.......
 Stato = fread (Buffer, 1,
     LenRec, Fil);
}
```

Il ciclo While è ripetuto fintantochè Stato è diverso da zero, quindi vuol dire che ci sono ancora record da caricare dal file. In questo ciclo viene richiamata la procedura **MostraRecord**, che visualizza un record del file, e quindi viene aggiornato il puntatore.

I) Alla fine del ciclo, le due istruzioni **if...** non servono assolutamente a niente, se non ad evitare che il compilatore C dia dei **warning**. Queste istruzioni possono essere omesse.

MostraRecord

Questa procedura serve per visualizzare, a gruppi di 16 alla volta, i 128 bytes corrispondenti al record del file.

A) La prima istruzione dichiara la variabile **Indice**, che quindi è locale, vista cioè solo all'interno della sola procedura.

B) Le due successive `PrintF` servono per preparare la maschera di visualizzazione. In questo caso, per visualizzare un numero esadecimale, si utilizza printf, con parametro **"%nnX"**, dove **nn** è il numero di digit (cifre) da visualizzare, ed **X** specifica che il numero deve essere visualizzato in esadecimale.

C) Il ciclo For richiama la procedura **Mostra16** per ogni riga di 16 bytes da visualizzare.

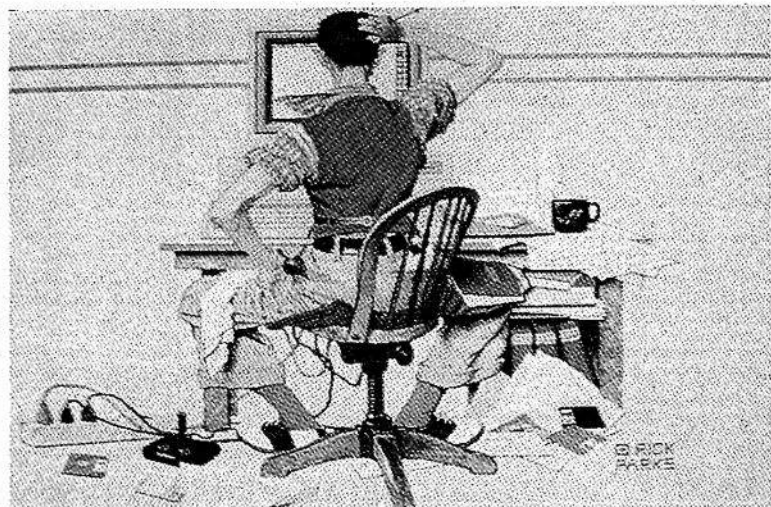
Mostra16

Questa procedura serve per visualizzare 16 bytes del file, sia in formato esadecimale che in formato ascii.

A) Anche in questo caso vengono dichiarate 2 variabili locali.

B) Printf serve per visualizzare l'offset corrente del file. Questa volta viene visualizzato in esadecimale ad 8 cifre. La **"l"** prima della X specifica che il parametro passato non è intero (2 bytes) ma intero lungo (4 bytes), appunto 8 cifre esadecimali.

C) Il primo ciclo For serve per visualizzare i 16 bytes del file in formato esadecimale a 2 caratteri (Vedere `printf... %02X`).

D) Il secondo ciclo For visualizza gli stessi 16 caratteri in ASCII. Se il carattere è fuori dai limiti visualizzabili (32 - 126) viene sostituito da un punto. L'istruzione `Putchar` serve appunto per visualizzare il carattere specificato.

di Domenico Pavone

Tanta memoria in più per nostra amata Amiga

Due drive un po' speciali e un disco rigido dal prezzo concorrenziale, con una digressione sulla Ram aggiuntiva per Amiga 500.

Dopo l'orgia di suoni e immagini che ha caratterizzato la "Vetrina" degli ultimi tempi, è d'obbligo una pausa di riflessione.

Da intendersi, è ovvio, non in senso strettamente letterale (qualche pagina vuota non soddisferebbe certo i lettori), quanto piuttosto come un ritorno alle esigenze primarie di **chi ha appena acquistato un Amiga** o comunque si accinge ad ampliarne il corredo hardware di base.

Detto in altre parole: drive supplementari e maggiore disponibilità di memoria. Voci, queste, che fino a non molto tempo fa costituivano una percentuale non indifferente dell'esborso necessario per disporre di un sistema Amiga... diciamo decente, per non parlare dell'esigua scelta quantitativa che il mercato proponeva (leggi: Commodore o niente).

I tempi, per fortuna, sono cambiati, e la disponibilità hardware si è allargata di molto, tanto da far sorgere un problema opposto: quello della scelta.

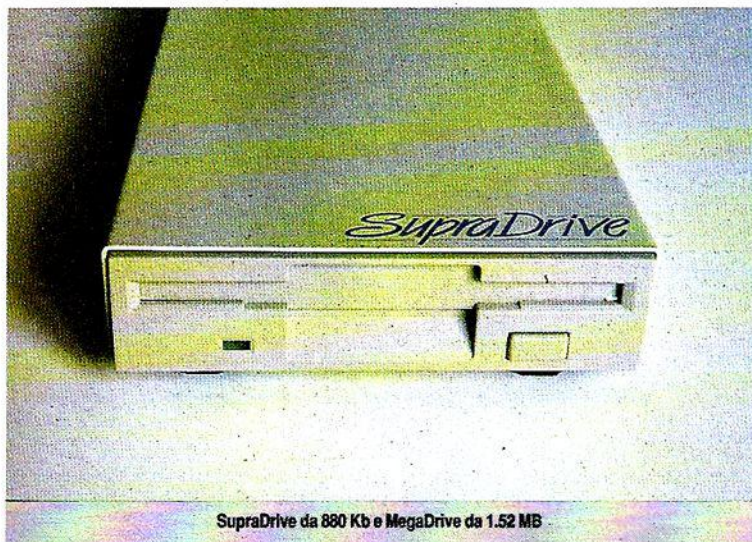
Fedeli ad una consuetudine che privilegia un rapporto prezzo/qualità accessibile ai più, eccoci quindi a prendere in esame alcune periferiche dal costo non esorbitante, ma con prestazioni particolari:

un floppy drive dotato di protezione contro l'attacco della maggior parte dei virus;

un altro che (udite, udite) consente di memorizzare su **disco 1.52 Megabyte** di dati;

una espansione di memoria per Amiga 500 che si caratterizza per la sua comodità d'uso, nonché per la quantità di ram liberamente inseribile: fino ad **8 Megabyte**!

Per chi, invece, dispone di un A-2000, e intende migliorare in modo decisivo la convivenza con il proprio computer, l'installazione di un **disco rigido** rappresenta un passo obbligato. Anche in questo settore la scelta è ampia, e una possibile è rappresentata dalla hard card da 42 Megabyte derivata dall'unione di un controller Supra Word Sync con una meccanica Fujitsu, a un prezzo davvero concorrenziale: 770.000 lire "chiavi in mano".

Vedremo tra breve se li vale.

SupraDrive da 880 Kb e MegaDrive da 1.52 MB

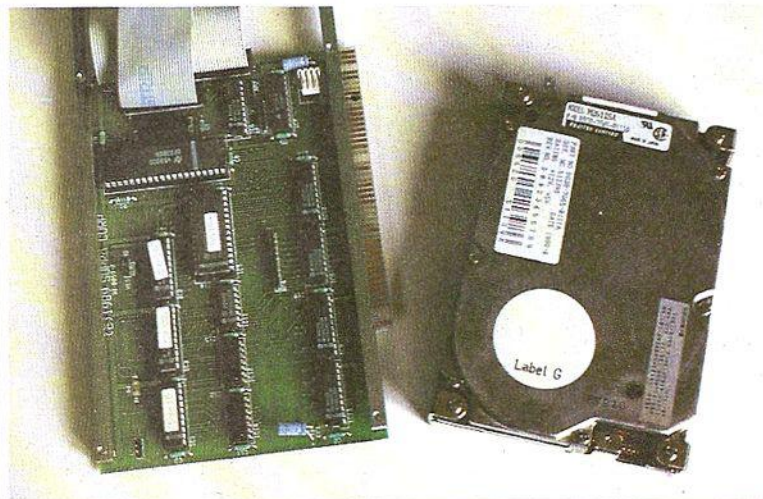
In vendita anche per corrispondenza presso
FLOPPERIA
Viale Monte Nero, 15
20135 - Milano
tel. (02) 55180484

SupraDrive

Questo drive per floppy disk, prodotto dalla ben nota **Supra Corporation**, all'apparenza non si discosta molto da molti altri consimili:

formato slim leggermente allungato, normalissimo frontalino con spia di attività e pulsantino di eiezione, cavo di connessione con spinotto dotato di comode viti di ancoraggio azionabili manualmente, anche senza l'uso di un cacciavite. Abbastanza consueta, ma per niente scontata, anche la **porta di espansione** sul posteriore dello chassis, che consente di aggangiare "in cascata" altri drive.

La compatibilità con Amiga è totale, nel senso che per installarlo non occorre effettuare altra manovra che l'inserimento del connettore alla porta drive di Amiga, quale che sia il modello posseduto: a seconda della configurazione hardware preesistente, diventerà la periferica **Df1:**, **Df2:**, eccetera.

Il primo elemento che colpisce, dopo l'installazione, è la silenziosità. Il che non vuol dire che da un drive ci si debba aspettare sinistri scricchiolii durante l'uso (anche se talvolta non mancano...), ma chi adopera Amiga con una certa intensità è certamente al corrente del famigerato "click" che caratterizza il funzionamento dei normali drive, cui di solito ci si oppone con artifizi software come il comando **Noclick** e similari.

Controller Supra e H/D prima dell'assemblaggio

Con Supradrive il problema è risolto radicalmente, in quanto l'hardware è provvisto di un "noise reduction" interno, in effetti molto efficace.

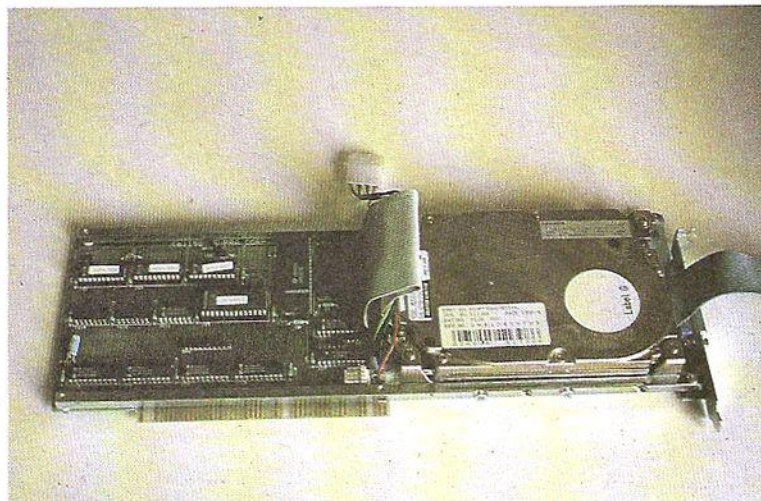
Ma la peculiarità del Supradrive è legata soprattutto ad un piccolo interruttore a tre posizioni posto sul retro dell'apparecchio. Banale, ma solo in apparenza.

Attraverso lo switch è infatti possibile non solo **disabilitare il drive** senza sconnetterlo fisicamente (posizione off), manovra notoriamente utile per risparmiare qualche Kilobyte di allocazione in memoria se si vive in... ristrettezze, ma soprattutto attivare una cosiddetta **modalità Protect**.

Questa, senza la necessità di alcun intervento esterno, inibisce totalmente l'accesso in scrittura al bootblock del floppy inserito, prevenendo in pratica ogni possibile **infezione dai virus** che si propagano attraverso il noto sfruttamento della traccia 0. Il meccanismo è talmente efficace da inibire addirittura anche la semplice formattazione di questo blocco, per cui, in questa eventualità, è necessario commutare lo switch nella normale posizione On.

Per testare la validità di questa feature, si è fatto un banale esperimento: lanciato il sistema con un floppy afflitto dal vecchio, ma sempre virulento, **Byte Bandit**, si è inserito un normale disco nel Supradrive impostato in modalità normale.

Poi, dopo aver agito sullo switch per inserire il Protect, se ne è inserito un altro. Poiché lo scopo delle manovre non nasconde intenti suicidi, si è poi spento il computer e lo si è riavviato con un floppy sicuramente "pulito". Breve parentesi: se vi venisse in mente di ripetere l'esperimento, non dimenticate dove riponete i dischi contagiati, potreste trovarvi alle prese con una strana epidemia.

Tornando in argomento: dopo aver lanciato il beneamato **VirusX**, questo se-

Hard Card assemblata (Supra controller e H/D Fujitsu)

gnalava immediatamente la presenza del virus nel primo disco, mentre il secondo ne risultava esente. Elementare quanto si vuole, ma il sistema funziona!

Qualcuno obietterà (a ragione) che esistono anche i cosiddetti link virus, in grado di associarsi direttamente ai files senza sfruttare la traccia 0 dei dischi, ma per questi sono necessari programmi appositi, per di più molto specifici. E Supradrive, nonostante tutto, rimane pur sempre un drive, non può certo sobbarcarsi del lavoro di tutto un computer.

Né, in fondo, lo si può pretendere da una periferica il cui prezzo si aggira sulle **160 Klire...**

Vista posteriore dell'Hrad Card per Amiga 2000

Mega-Drive

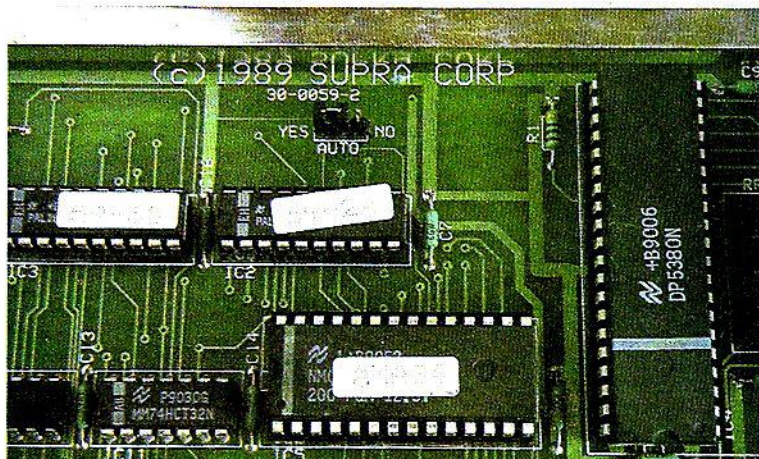
Forse più adatto a chi almeno due drive, per così dire normali, già li possiede, ecco un prodotto in grado di colmare una lacuna finora evidente nel mondo Amiga.

Già da tempo, infatti, l'universo "parallelo" dei Pc Ibm e compatibili usufruisce di una risorsa non indifferente, ovvero la possibilità di memorizzare su floppy da 3.5 pollici una quantità di dati praticamente doppia rispetto a quanto era prima consentito: 1.44 Megabyte invece dei soliti 720 KB.

Amiga, nonostante il passare del tempo, sembrava confinata ai suoi angusti (si fa per dire) 880 Chilobyte, ma la necessita, com'è noto, aguzza l'ingegno. Dove (almeno finora) non arriva il sistema, ecco che provvede una periferica e il suo software a corredo.

Già, perché Mega-Drive, ad un prezzo solo di poco superiore ai comuni drive (250.000 lire), permette la formattazione di supporti ad alta densità per un totale di **1.5 Megabyte disponibili**! Un risultato ottenuto non solo via hardware, ma anche grazie alla realizzazione di un apposito dispositivo software che si sostituisce alla normale gestione della **Trackdisk.device**. Niente paura: anche se siete semplici user di Amiga, l'installazione del tutto è molto semplice e automatizzata.

Prima di dare un'occhiata più approfondita al drive, è opportuna una breve precisazione. Si è già detto che, come del resto avviene per i "cugini" *msdos*siani, per supportare una simile capacità è indispensabile adoperare floppy disk ad alta densità (**Hd**), non i comuni Double Side / Double Density (**Ds-Dd**). Per garantire affidabilità alla grossa quantità di dati memorizzabili con Mega-Drive, è necessario aggiungere che tali dischi devono essere **realmente** 100% "free error", ovvero floppy di marca garantita. Ad una bassa qualità dei supporti, di solito riscontrabile nei più economici (anche se hd!), corrisponderà un sicuro incorrere, prima o poi, in una fatale segnalazione **Read/Write Error** del nostro Amiga.

Esternamente, Mega-Drive non presenta caratteristiche particolari, mostrandosi in tutto e per tutto simile ad un qualunque altro drive esterno. E' dotato di **interruttore** per escluderne, eventualmente, l'influenza sul sistema (più che altro se si desidera risparmiare memoria senza sconnetterlo), di un **connettore**

Si noti il ponticello per la scelta dell'autoboot

standard da inserire nella porta drive del computer, e di una **presa a 23 poli**, anch'essa standard, per l'eventuale collegamento di altri drive.

Una volta connessa, la periferica non viene automaticamente rilevata dal sistema, per cui è necessario che sia "mountata". L'operazione, da un punto di vista pratico, non presenta alcun problema: è sufficiente attivare Amiga con una copia del disco Workbench, inserire il floppy fornito a corredo del drive, ed accedervi da Workbench come di consueto. Al suo interno sono presenti due icone, a nome **Mega2** e **Mega4**. Se si possiede un Amiga 500 (o 1000) occorrerà biclickare su Mega2, per Amiga 2000 è invece da adoperare Mega4. Tutto qua.

Un reset al sistema, e la nuova periferica sarà attivata, come dimostrato da un eventuale comando Info da ambiente Shell o dalla consueta comparsa di un'icona disco da Workbench, sempre che un floppy sia inserito nel drive. Nel primo caso, l'indicazione relativa a Mega-Drive corrisponderà al device **Hf1:**, che sarà necessario specificare in tutte le normali attività Dos, esattamente come si farebbe per qualunque altra unità (df0:, df1:, eccetera).

Per i non esperti, va ricordato in modo particolare di effettuare l'installazione adoperando una **copia** del proprio disco di boot (comunemente il Workbench), dal quale, magari, eliminare qualche programma superfluo (il Notepad, per esempio) per creare spazio supplementare. L'installazione, infatti, opera delle modifiche sulla startup-sequence (per gli smanettoni: aggiungendo un comando **Mount Hf1:**) ed alla **mountlist**, copiando poi nella directory **Devs** di sistema uno specifico file di gestione. Ai possessori di A-2000, se qualcosa non dovesse funzionare a dovere, si consiglia di provare ad impartire da Shell un comando...

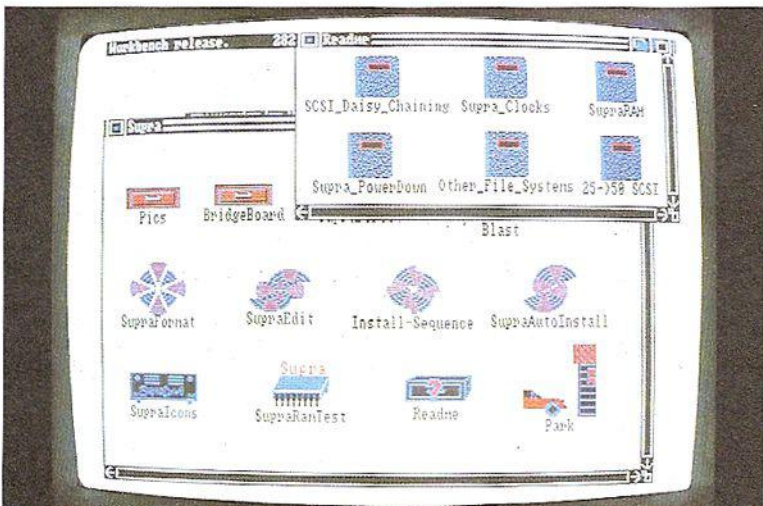
Il connettore SCSI consente l'utilizzo di altre periferiche

```
Copy Gcr:mega19t.device
Devs:gcrdisk.device
```

Dopo un reset, tutto dovrebbe essere in regola.

Eseguita (una tantum) la procedura di installazione, il drive andrà adoperato senza particolari attenzioni dal punto di vista gestionale. Il normale **Format** (da Shell) o la relativa selezione da menu del Workbench inizializzeranno senza problemi i dischi inseriti in Mega-Drive, naturalmente in accordo con la nuova capacità di 1.5 MB. Lo stesso dicasi per qualunque operazione di caricamento, salvataggio o copia di files, con unica eccezione per il comando **Install**, che non agirà sul nuovo formato; cosa del resto ovvia in quanto un bootblock, anche se fosse installabile, non verrebbe comunque riconosciuto fisicamente dal sistema.

La maggiore capacità dei floppy, infatti, è legata all'incrementato numero dei blocchi per settore sul disco, che diventano **19** invece dei tradizionali **11**.

Ma tutte queste non sono che informazioni supplementari per eventuali smanettoni, di nessuna importanza per chi voglia solo ritrovarsi finalmente con tanto, tanto spazio in più su disco. E con una nuova precauzione cui badare: se un backup è utile già con i soliti 880 Kb a disposizione, con 1 mega e mezzo di dati diventa quasi un obbligo. Magari adoperando per la copia un paio di floppy del-

S/W di gestione dell'hard disk

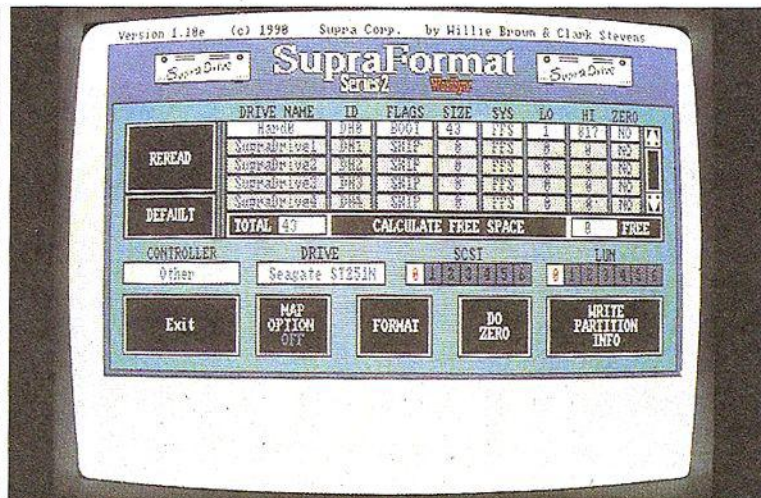
l'ormai "vecchio" formato... Inutile precisare che, volendo scambiar dischetti tra amici, dovrete accertarvi, prima dello scambio, che il destinatario del dischetto possegga anch'egli un Mega-Drive.

SupraRam 500-RX

Espandere Amiga, soprattutto quando si parla del modello 500, più che un optional è ormai diventato un obbligo. Proprio per questo motivo, anche in questo settore l'evoluzione tecnica ha fatto passi da gigante, sfatando tra l'altro il luogo comune che vuole Amiga 500 relegato al misero ruolo di game-machine.

Se i miseri 512 KB di ram in dotazione base si avviano a scomparire, lo stesso processo si può dire avviato nei confronti di quella che fino a poco tempo fa era una scelta obbligata: l'espansione di altri 512 KB da inserire nel cassetto inferiore del computer.

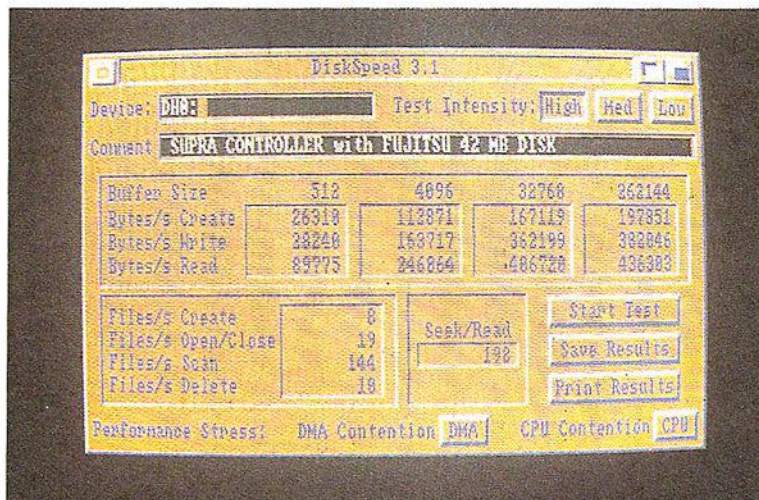
Chi proprio pensa che non utilizzerà mai Amiga per scopi meno che ludici può sempre accontentarsi di questa (farne totalmente a meno è decisamente impensabile anche per un gamofilo ad oltranza), ma la strada più pratica è ormai diventata un'altra.

Esistono, è vero, alcune espansioni che, sfruttando lo stesso bus di connessione, giungono anche fino a 2 megabyte, ma... e se un domani si tornasse ad avere fame di memoria? O si volesse dotare il nostro/vostro beniamino di una scheda di emulazione Ms-Dos? Ahinoi (o meglio, ahivoi), non resterebbe che gettare alle ortiche l'espansione.

Ecco allora che la soluzione più ovvia consiste nell'acquistare un tipo di scheda liberamente configurabile e che sfrutti lo slot laterale di A500, lasciando libero il famoso "cassettino" per eventuali altri scopi. Detto fatto, eccoci a parlare dunque di Supraram 500RX, uno dei prodotti più raffinati del genere, che in pratica può soddisfare ogni tipo di esigenza, tanto hardware che economica.

Si tratta, infatti, di un telaio metallico dalle dimensioni ultra ergonomiche (2.5 cm. di spessore per 25 cm di lunghezza), fornito di un connettore solidale alla sua base che si inserisce nel "pettine" posto sul lato sinistro di A-500, normalmente nascosto da una copertura plastica. Una volta effettuata la connessione (con la dovuta delicatezza), l'espansione fa corpo unico con la tastiera del computer, senza creare il minimo problema e senza la necessità di alimentazione esterna.

Qualcuno obietterà che, usufruendo di questa espansione, si esclude la possibilità di aggiungere un hard disk o qualcosa come la cartuccia Action Replay, che normalmente sfruttano lo stesso bus. Ebbene, **non è così**.

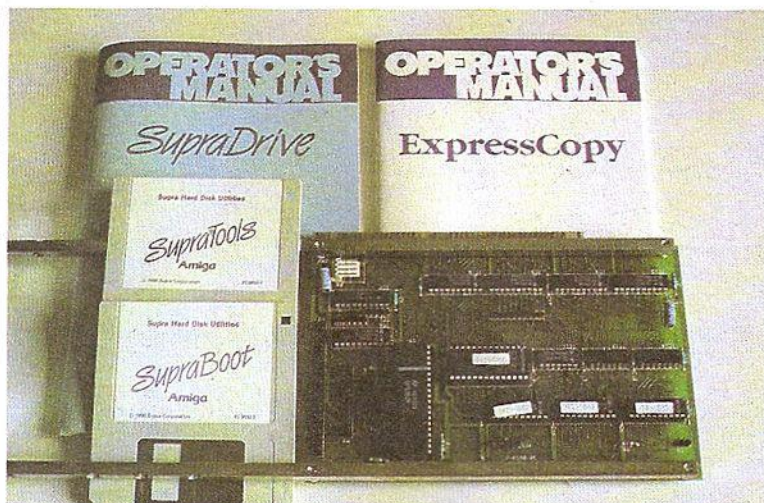
La Supraram, infatti, nella sua controfaccia laterale, presenta uno sportellino analogo a quello del computer, cui si accede rimuovendo due viti che lo mantengono serrato. Qui, per la gioia dei patiti cinquecentisti, è disponibile lo stesso slot di Amiga 500, per un totale ed opportuno "pass-through". Se quindi si dispone (o si prevede l'acquisto) di un hard disk magari non dotato della stessa opportunità, non sorgerà alcun proble-

Supra Format, il programma di gestione dell'hard disk

Risultato del test effettuato con DiskSpeed

ma: basterà sistemarli uno accanto all'altro, con l'espansione inserita per prima. Tra l'altro, proprio in questa eventualità, non sarà necessario ricorrere a dischi rigidi con controller dotato di ram aggiuntiva, ammortizzando così buona parte del prezzo dell'espansione.

Questa, come già accennato, può aggiungere alla dotazione di memoria del computer fino a **8 Megabyte di Ram**, ma può benissimo essere acquistata con "a bordo" **solo 1 Mb**, rimandando a tempi migliori ulteriori ampliamenti. L'installazione fisica della ram risulta molto semplice: è sufficiente rimuovere il contenitore esterno ed innestare negli zoccoli il set di chip desiderato, a seconda della configurazione cui si aspira.

Fino ad un massimo di 2 Megabyte è infatti possibile adoperare gruppi di 4, 8, oppure 16 chip da 256 Kbit x 4, mentre per configurazioni di 2, 4 oppure 8 Megabyte sarà necessario ricorrere allo stesso ammontare di chip da 1Megabit x 4. Al momento dell'acquisto l'espansione è già configurata in base alla ram di cui dispone, mentre nel caso di aggiunte future può rendersi necessario modificare lo stato di **tre Jumper interni**, operazione peraltro facilitata da chiari schemi illustrati nel manuale fornito a corredo.

La configurazione con la quale la si acquista, influisce come ovvio in misura determinante sul prezzo, che può dunque oscillare di parecchio. Si pensi però, come punto di riferimento, che con **275.000 lire** si può accedere a Supraram con 1 Mega già installato, mentre in configurazione doppia si raggiungono le 399.000.

Manualistica e s/w allegati al Supra Controller

Nel caso di particolari esigenze, Supraram dispone anche di un piccolo interruttore a levetta sul posteriore, che consente di escludere la memoria aggiuntiva dal sistema, senza dover rimuovere fisicamente l'espansione. Un altro switch, questo interno allo chassis ma ugualmente raggiungibile dall'esterno attraverso un foro, permette inoltre di attivare un modo "test". In pratica: una volta attivato Amiga con un floppy fornito assieme all'hardware, il modo Test consente un controllo totale sui chip installati, che in caso di malfunzionamenti possono essere identificati con precisione.

Tutto, insomma, è curato nei minimi particolari, fornendo la gradevole impressione di avere a che fare (una volta tanto) con qualcosa di professionale, pur se legata al più "piccolo" degli Amiga.

Espansione laterale per A-500; si noti il connettore

Un hard disk per A-2000

Nella magior parte dei casi, chi possiede un Amiga 2000 ha optato per questo modello in funzione della sua facile ed ampia espandibilità, in attesa magari di veder fiorire (non proprio per grazia ricevuta) tante belle schede nei suoi slot interni.

La prima a fare la sua comparsa, di solito, è un'espansione di memoria, ma l'elemento che realmente "fa la differenza" è l'hard disk.

La scelta, inutile dire, è molto vasta. E, tanto per cambiare, legata strettamente al fattore economico, oltre che a esigenze specifiche. Un giusto compromesso

tra qualità (nonché capienza e velocità) e prezzo non è facile da definire, ma per un utenza media ecco una soluzione che può senz'altro considerarsi vantaggiosa: un hard disk da **42 Megabyte** al prezzo di **770.000 lire**, ovviamente **controller compreso**.

Chi fosse digiuno della terminologia legata a questo tipo di periferiche può andarsi a spulciare il **n. 79** della rivista; gli altri converranno sicuramente che si tratta di un costo decisamente contenuto.

E, c'è da precisare, non a scapito delle prestazioni, garantite da un controller Supra Word Sync di prima qualità, associata ad una **meccanica Fujitsu M2611SA da 3.5"**.

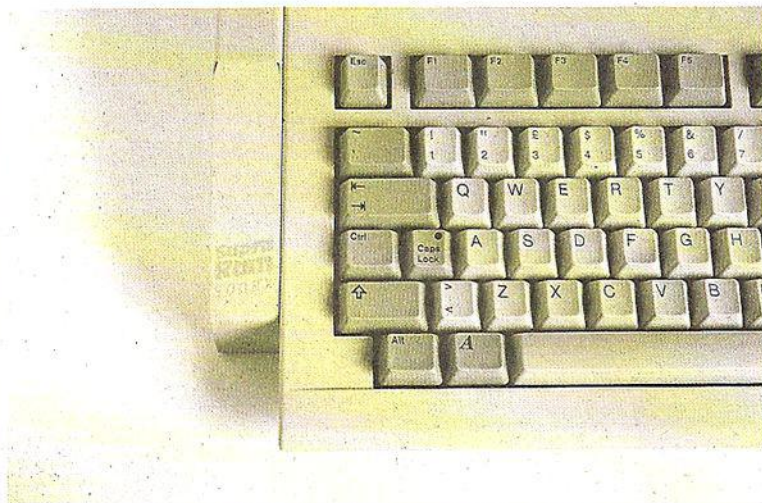
I due elementi (controller e disco rigido) vanno assemblati in un'unica hard card da inserire in uno degli slot di Amiga, con poche e semplici manovre illustrate molto dettagliatamente nella manualistica a corredo, sulle quali è quindi inutile soffermarsi.

Piuttosto, c'è da segnalare lo **standard SCSI** di cui è dotata la scheda, ormai divenuto un obbligo qualitativo, che consente tra l'altro di collegare fino a **7 dispositivi** dotati di tale interfaccia. Allo scopo, è anche disponibile un connettore esterno, raggiungibile sul posteriore del computer dopo l'installazione della hard card.

Il controller è fornito di opportuna **Rom** per procedere all'**autoboot** di Amiga, requisito pressoché indispensabile per un uso realmente comodo, naturalmente implementabile solo se si dispone del sistema operativo **1.3**.

Anche in questo caso, comunque, è possibile disabilitarlo in qualunque momento modificando lo stato di un ponticello (jumper) presente sul circuito stampato della scheda.

Il controller è dotato di una buona gestione degli accessi **Dma** (Direct Memory Access) con rallentamenti minimi del sistema multitasking, e una ottima velocità di invio dei dati al computer garantita dalla trasmissione di una intera word per volta (due byte). La reale velocità operativa, non va dimenticato, dipende molto anche dalla meccanica adottata, che naturalmente può essere scelta diversamente da quanto qui proposto, in accordo con le proprie esigenze e disponibilità economica.

L'accoppiata Supra/Fujitsu 42 Mbyte, sempre con un occhio al prezzo, è risultata comunque decisamente affidabile e performante (questa la rubiamo alla formula 1), soprattutto in lettura, come dimostrato dal benchmark eseguito con il programma **DiskSpeed** e riprodotto nelle foto di queste pagine.

Notare il minimo ingombro richiesto da Supra Ram

Per una gestione ottimale dell'hard disk la scheda viene fornita con una dotazione notevole di manualistica e di software.

Quest'ultimo, in particolare, si caratterizza per la sua estrema facilità di impiego, prendendo per mano l'utente anche più inesperto sin dai primi passi, facilitandogli ogni operazione con svariate utility applicative.

La prima, fondamentale, è **Supra Format**, che consente di inizializzare il disco scegliendo con semplici tocchi del mouse le caratteristiche da assegnargli, come per esempio l'eventuale divisione in più partizioni, il loro nome e dimensione, e a quale di queste attribuire il compito di "bootare" il sistema.

Ma anche, se non soprattutto, il tipo di file system sotto il quale operare: **FFS** (**F**ast **F**ile **S**ystem, che può essere adoperato anche per la partizione di boot) e **Old** (il normale **A**miga **F**ile **S**ystem), ma nulla vieta che si adoperino altri standard come **Ms-Dos**, **Unix**, o **Macintosh** se si dispone degli appropriati emulatori hard/software.

Altri file consentono poi di facilitare l'installazione software dell'unità, che va opportunamente "montata" nella sua startup - sequence.

A questo proposito, tra l'altro, è disponibile un singolo comando **Supramount**, che provvede da solo a rendere accessibile al sistema tutte le partizioni, senza la necessità di impartire più comandi.

Automatizzata è anche la procedura di trasferimento nelle opportune directory della **Boot Partition** (che può anche essere unica e comprendere tutti i 42 Mega dell'hard disk) di alcuni file indispensabili al sistema per dialogare correttamente con l'unità, come l'**Harddisk.device**, o il **FastFileSystem**.

Non manca poi il solito programma per "parcheggiare" le testine del disco in caso di trasporto, e una Directory Utility di antica tradizione, il **Climate**.

Molto utile, infine, **Express Copy**, un software che consente di effettuare il backup completo dell'unità su (molti) floppy disk in modalità totalmente automatica e con una segnalazione preventiva sull'esatto numero di floppy necessari per svolgere l'operazione.

In definitiva, si può esprimere per questa hard card un giudizio decisamente positivo, soprattutto in rapporto al prezzo con il quale viene proposta.

Per prestazioni superiori, del resto, altre scelte non mancano: tutt'al più, a mancare, possono essere i numerosi biglietti da centomila da aggiungere al costo della Supra/Fujitsu...

di Luigi Callegari

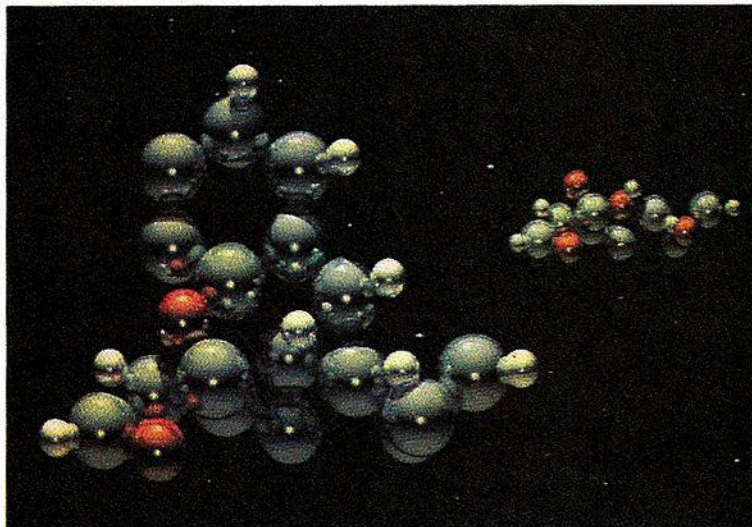
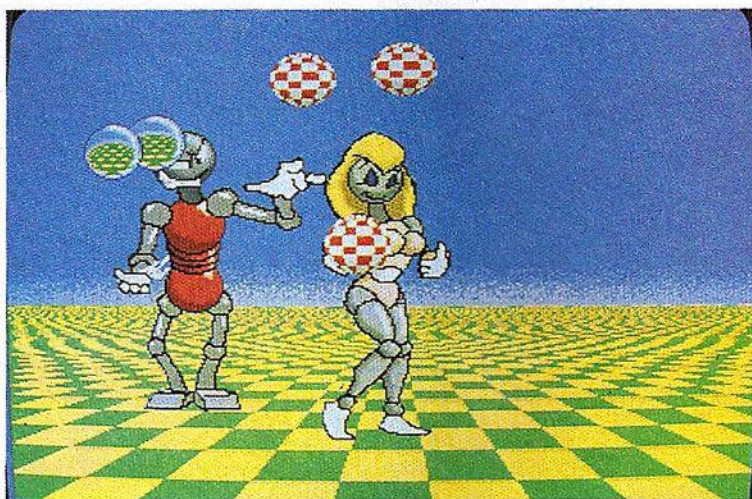
Bit Movie 1991

Una manifest azione in cui la grafica era la protagonista indiscussa; e con essa, naturalmente, Amiga...

Bit.Movie è una mostra nazionale dedicata alle applicazioni artistiche realizzate con il calcolatore, giunta quest'anno alla quarta edizione.

L'edizione 1991 si è tenuta nel Palazzo del Turismo di Riccione, tra il 25 ed il 28 aprile, in Piazzale Ceccarini, cuore allegro e pulsante di questa ben nota località dell'Adriatico, conosciuta solitamente più per la vita mondana che per le manifestazioni culturali, spesso prestigiose, che vi si tengono.

La scorsa edizione contò circa 2000 visitatori, all'incirca raddoppiati quest'anno.

Difatti al Bit.Movie vi sono sempre più cose interessanti da seguire: il concorso per la migliore **animazione grafica** in tempo reale su personal computer, la sezione **musicale** con concerti e spiegazioni sull'uso dei PC in campo artistico,

la sezione **laboratori** con seminari sull'uso di vari pacchetti molti diffusi per l'Amiga (Deluxe Paint, Professional Page, Imagine, Broadcast Titler), una **mostra fotografica** di immagini prodotte da computer di varia potenza (dall'Amiga 500 ai mini dedicati usati da softhouse specializzate americane, generati col lavoro di decine di artisti e programmatori), un sistema **multivisione** che presentava a ciclo continuo immagini e musiche inerenti l'informatica, una sezione **video** ove si proiettavano animazioni grafiche prodotte da alcune delle softhouse più famose del mondo (con filmati provenienti in esclusiva dal festival IMAGINA di Montecarlo, forse la più famosa manifestazione mondiale di animazioni al computer) e la sezione **didattica** che trattava dell'uso di Amiga nella scuola dell'obbligo.

Il concorso

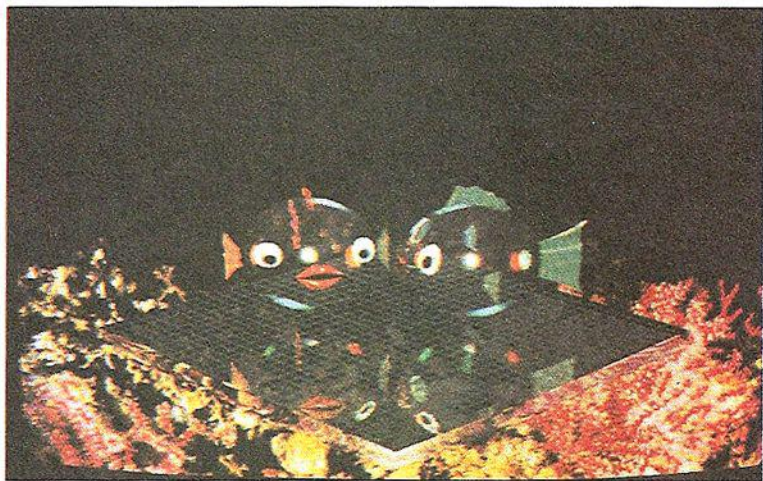
Il bando del concorso per la migliore animazione grafica su personal computer (Amiga, MS-DOS, Macintosh...) è stato pubblicato da molte riviste italiane e straniere (tra le quali, Amiga World, Amazing Amiga e Amiga User International).

Gli organizzatori hanno dunque potuto contare su circa **200 lavori**, fra i quali ne sono stati selezionati una trentina da sottoporre a ciclo continuo al pubblico che visitava la mostra (fornito di un apposito modulo per la votazione all'ingresso) e ad una giuria specializzata.

Il montepremi di quest'anno era di ben 1,5 milioni, 1 milione e cinquecentomila lire per il primo, il secondo ed il terzo classificato nelle due categorie (pubblico e giuria).

Le opere vincitrici sono state per il pubblico **Glass Fish!** di Mrsek Giuseppe Milko da Vobarno (BS), **The Dating Game** dell'americano Schwartz e **Chess** ancora di Mrsek G. Milko (già vincitore della precedente edizione del Bit.Movie con **Waterchess**).

Per la giuria, composta da sei persone qualificate scelte nel mondo della computer grafica e dell'informatica, l'ordine di arrivo è stato invece **The Dating Game** di Schwartz, **Glass Fish!** di Milko e **Musique** di Stefano Piloni.

Alcuni fotogrammi delle migliori animazioni sono visibili in queste pagine. Da notare che quest'anno **29** delle trenta

animazioni patecipanti alla fase finale erano state realizzate con **Amiga** ed **una sola** con **Macintosh**. Dal momento che le animazioni, a norma di regolamento, dovevano essere riproducibili in tempo reale davanti al pubblico (quindi **non** da videocassetta), è logico che l'enorme velocità grafica di Amiga ha prevalso facilmente su sistemi come il Mac e, soprattutto, Ms - Dos con schede VGA (estremamente lenti se usati in tempo reale).

Oltretutto, dei sei premi, quattro sono stati vinti da stupende e iper-realistiche animazioni prodotte in ray-tracing, mondo tuttora vietato e lontano per la lentezza e la magra varietà cromatica delle schede VGA (super od ultra), in confronto con i 4096 colori contemporanei di Amiga e la velocità operativa del suo coprocessore grafico e del Blitter.

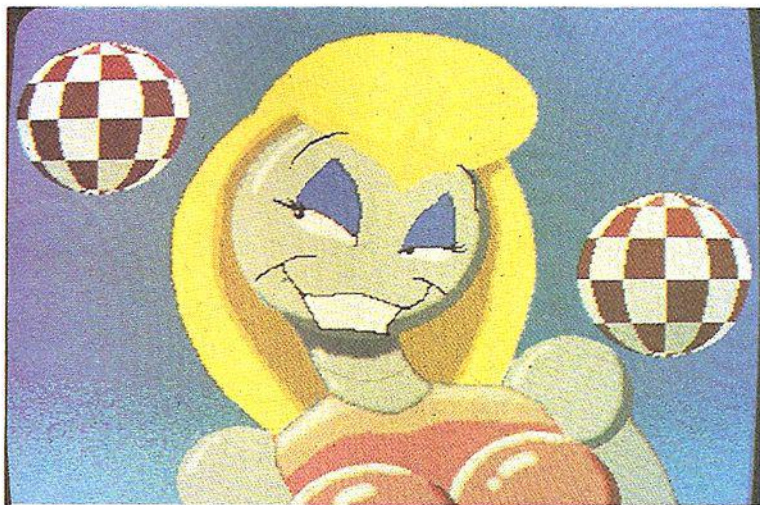
I corsi

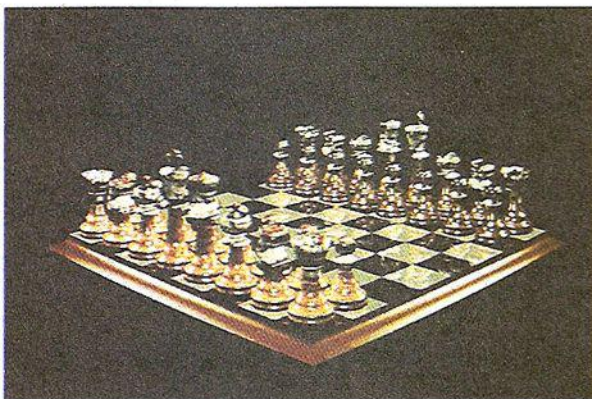
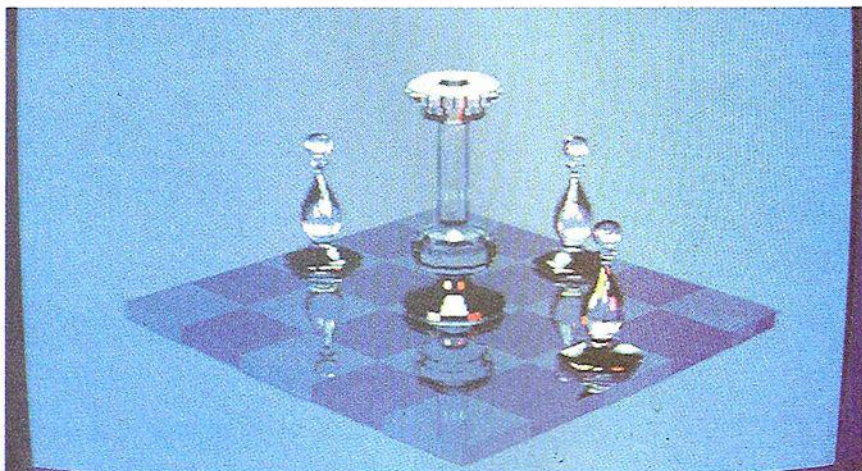
Ad un pubblico ristretto, ma estremamente interessato, sono stati illustrati da persone esperte i misteri di alcuni pacchetti applicativi per Amiga.

I corsi, che (secondo programma) dovevano durare solo un paio di ore al giorno, si sono invece regolarmente protratti per ore e ore, durante le quali iscritti e curiosi hanno potuto studiare ed apprezzare le capacità, a volte esclusive, offerte dal migliore software in circolazione per Amiga. Dal glorioso Deluxe Paint III al nuovissimo tool di generazione grafica ray-trace "Imagine", tutti i pacchetti hanno goduto di un buon riscontro tra il pubblico.

Nella sezione musicale, si è dimostrato come si possano usare piccoli sistemi computerizzati, come gli **Atari ST**, con apparecchiature MIDI professionali. Mentre, al mattino, gli organizzatori spiegavano agli interessati in un vero e proprio corso, come si usavano pacchetti software ed attrezzature, durante la serata gli stessi tenevano, con le attrezzature trattate nel seminario, degli apprezzati concerti seguiti da una folla di ascoltatori che non hanno lesinato applausi al **Fusion Quartet** di Stefano Villani.

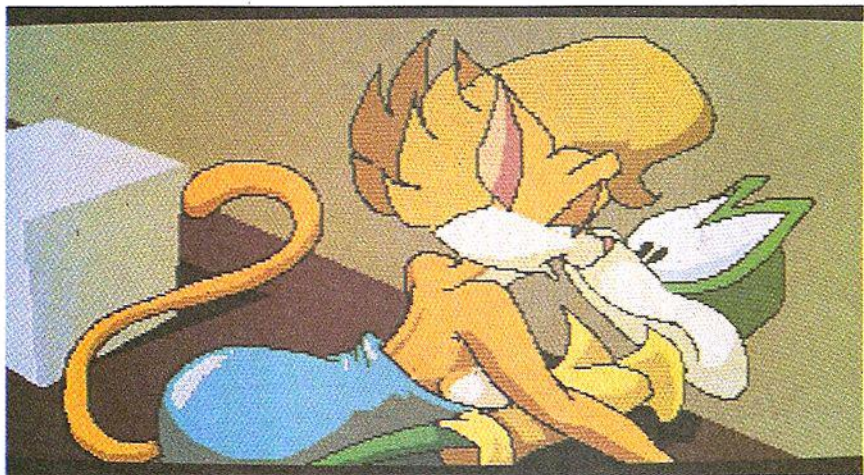
Nella sezione didattica la attivissima professoressa Germana Pellegrini, no-

minata Cavaliere del Lavoro per la sua carriera di insegnante, ha dimostrato come si possa usare efficacemente Amiga come strumento di sviluppo della creatività artistica di giovani e giovanissimi utenti. Abbiamo potuto assistere, infatti, alla realizzazione e rielaborazione personale di un semplice cartone animato con Deluxe Paint III da parte di allievi di scuole elementari, con tanto di suono e di parlato, con trasferimento finale su VCR. Poi all'uso dei (relativamente) economici e pregevoli digitalizzatori audio e video (Digiview) di Amiga per trasformare immagini di artisti e pittori in "screen" del Deluxe Paint da rielaborare e studiare con l'ausilio del computer.

Il futuro

Il Bit.Movie è sicuramente una manifestazione che guadagna sempre maggiore importanza. Prodotta con l'ausilio dell'Assessorato alla Cultura e dell'Assessorato al Turismo di Riccione, quest'anno è stato presentato oltre che su vari quotidiani e TV locali, anche in alcune trasmissioni della RAI (TG3 e "Uno Mattina").

Anche il numero elevato di visitatori dimostra che, a volte, la buona volontà di alcune persone, gli organizzatori, può produrre risultati paragonabili a manifestazioni di grande richiamo all'estero, impossibili da realizzare senza l'aiuto di sponsor.

La prossima edizione del Bit.Movie sarà ampiamente pubblicizzata su parecchie riviste del settore, compresa la nostra.

Sarà quindi facile, per chiunque voglia partecipare, sapere quali sono le modalità di partecipazione ed invio dei propri lavori.

Chi volesse, comunque, aspirare ad un premio, è bene che inizi da subito a realizzare elaborazioni sul proprio Amiga, in modo da studiare tecniche di gestione di animazioni (visti i tempi di elaborazione dei programmi "ray-trace" offerti dalla... concorrenza), specialmente se si possiede un normale Amiga non velocizzato!

di Gregor Samsa

Due cuori e una grafica

Adoperare una stessa immagine su Amiga e Pc può sembrare un'impresa impossibile, eppure...

Nonostante l'aspra diatriba che spesso divide gli utenti dei sistemi Amiga e quelli del cosiddetto "universo" Ms - Dos, su un elemento si troveranno entrambi sicuramente d'accordo: anche per applicazioni non strettamente grafiche, l'esibizione sul monitor di una gradevole immagine multicolore non può che rendere accattivante qualunque tipo di prestazione, professionale o meno che sia.

Se questo può essere dato per scontato nell'ambiente Amiga, maggiormente predisposto ad ogni tipo di espressione grafica, spesso viene sottovalutato da chi ancora ha un'idea dei pc compatibili legata al classico color verde (o ambra) di un monitor monocromatico. L'evoluzione tecnica, ma soprattutto l'abbassamento dei prezzi delle varie schede grafiche **Vga** e **Super Vga**, ha ormai reso obsolete simili idee, aprendo la strada a risoluzioni sino a qualche tempo fa impensabili, e molto vicine a quelle riscontrabili su Amiga. Detto questo, va riconosciuto comunque che, ad un livello medio/basso di costi, Amiga dispone in effetti di un gran numero di tools software che consentono facili ma sofisticate manipolazioni grafiche.

No, non si intende con questo riaccendere la sterile polemica tra i fan dei due sistemi, né "fare il tifo" per uno o l'altro. E non si ignora che esistono economici digitalizzatori per entrambi i computer, o addirittura programmi in versione "doppia" (Pc e Amiga) come il **Deluxe Paint**. Piuttosto, si vuole far riflettere concretamente sulla possibilità di sfruttare entrambi i mezzi per raggiungere risultati migliori.

Detto in altri termini: quanti ms-dossiani (magari solo per lavoro) hanno in casa un Amiga 500 anche solo per giocarci, o per il giovane figlio (ottima scusa...)? E quanti amighi, di contro, non hanno rapporti con "l'altro mondo", magari solo attraverso i diffusissimi emulatori hartd/soft, non foss'altro che per non precludersi settori di conoscenza che un domani potrebbero servire?

Retorica a parte, per tutti costoro (e per gli smanettoni incalliti) può risultare proficuo, o anche solo sfizioso, trasferire elementi grafici (leggi: schermate, ritagli, clips, eccetera) da un ambiente all'altro, ovviamente in modo che risultino utilizzabili in entrambi. Per gli utenti Amiga può essere vantaggioso acquisire immagini immagazzinate nell'hard disk di un Pc (o, perché no, da una **bbs**), mentre gli altri potranno trarre il vantaggio inverso, o magari adoperare Amiga per i ritocchi e le elaborazioni del caso, per poi ritrasferire il tutto in un professionalissimo Ventura.

La cosa non è complessa, purché... se ne conosca il modo (ah, Lapalisse!).

Vediamo dunque quali problemi possono sorgere, e soprattutto con quali strumenti software risolverli.

Le schermate grafiche sono sempre molto belle a vedersi

Simile e dissimile

Intanto viene dato per scontato che si sia a conoscenza delle modalità di trasferimento fisico di un file da Amiga a Pc e viceversa.

L'argomento è stato trattato più volte sulla nostra rivista (si veda per esempio il **n. 84**), per cui, senza soffermarci più di tanto, ricorderemo che esistono pro-

grammi per Amiga (su Pc **non** ne circolano) atti a copiare un file su dischi dell'altro formato (**Dos2Dos**, **Cross Dos**, **Messy Dos**); in alternativa, si possono connettere i due computer attraverso la porta seriale, e trasferire i file tramite comuni programmi di comunicazione, solitamente adoperati per i modem. Quale che sia la tecnica o il software adottato, il file va ovviamente "esportato" **integro**, ovvero senza attivare opzioni particolari come quella riservata alla conversione ASCII.

Fatta questa premessa, si possono cominciare a esaminare (almeno per ora, superficialmente) quegli aspetti che fanno realmente differenza tra i due modi di gestire la grafica. Il primo che ci si può aspettare, riguarda la **risoluzione** di una immagine, e il **numero di colori** in essa presenti.

Il problema, in realtà, interessa maggiormente il trasferimento (e l'utilizzo) di una immagine da Amiga a Pc, in quanto quest'ultimo ambiente è fortemente condizionato dal tipo di scheda grafica che monta.

Ma può anche essere vero il contrario, se si prendono in considerazione elementi grafici finalizzati ad un uso associato (p. es.) a Super Vga da 1024 x 768 pixel con 256 colori. Creato il problema, eccoci subito a ridimensionarlo. La risoluzione in pixel, nell'ottica di un trasferimento, va vista anzitutto in funzione del software che dovrà sfruttarla. Se, per esempio, si intende adoperare (in uno qualunque dei due computer) un visualizzatore di immagini che prevede l'uso di risoluzioni maggiori, magari con la possibilità di farle "scrollare" in *overscan*, il problema non sussiste. Se, invece, si pensa a qualcosa che andrà necessariamente contenuta in una singola videata, allora occorre prestare attenzione al fattore risoluzione. Tuttavia, visto l'elevato numero di convertitori esistenti, e dei quali tra breve approfondiremo la conoscenza, la cosa non presenta particolari difficoltà.

Un'immagine Ham di 320 x 200 in 4096 colori (Amiga)

Chiaro che, se per esempio si intende trasferire una schermata di Amiga con risoluzione 640 x 512 (tipicamente in grafica interlacciata), occorrerà non dimenticare che l'equivalente risoluzione in ambiente Pc sarà di 640 x 480 pixel, ma potrebbe anche risultare obbligato un formato inferiore (a seconda della scheda): in altre parole, occorrerà prevedere l'uso di programmi di conversione che rendano possibile un aggiustamento del genere.

Le stesse considerazioni valgono per il colore, forse ancora più "delicato" da trattare, e per il quale non si può indicare una regola generale.

Pochi problemi, è ovvio, per risoluzioni che implementano pochi colori, sempre che in ambiente Pc sia presente una scheda grafica adeguata; in caso contrario si renderà necessario anche in questo caso qualche intervento di modifica. Qualitativamente critico può risultare poi

AMIGA

NOME	Show	Ham	scelta risol. pixel	scelta numero colori	NOTE
ImageLink	NO	SI	SI	SI	Traduce Iff, Gif, Pcx, Tiff, ecc.
Transfer24	SI	SI	SI	NO	Solo in ambito Iff
Hamsharp	NO	SI	NO	NO	Traduce da Gif a Iff
Iff2gif	NO	NO	NO	NO	Traduce da Iff a Gif
Hamgif	SI	SI	NO	NO	Solo visualizzatore Gif
Virtgif	SI	SI	NO	NO	Mostra Gif anche in overscan
Showgif	SI	NO	NO	NO	Solo visualizzatore Gif

I programmi di "conversione" più diffusi in ambiente Amiga

il cosiddetto **modo Ham** di Amiga, in grado di sfruttare una gamma cromatica superiore ai **256 colori**, e quindi mediamente al di sopra delle capacità "**pc**ine" di rappresentazione. Ma il problema si riduce ancora all'uso di software adeguato, operando delle modifiche preliminari in ambiente Amiga.

Quali standard...

E siamo giunti così al vero nocciolo della questione: la codifica del formato. O meglio, vista dall'ottica del computer, la **decodifica**.

Tutte le informazioni che riguardano una immagine grafica, colore, definizione, e rappresentazione della cosiddetta **bitmap** (la descrizione vera e propria del disegno in forma digitale), possono essere memorizzate in diverso modo, a seconda del software che le crea. Esistono, è vero, degli standard di fatto, ma in rapporto a quanto stiamo trattando in effetti non ce n'è uno realmente "universale". Su Amiga, per esempio, lo standard più diffuso è l'**IFF ILBM**, ma altre forme di espressione grafica per lo più tridimensionale sfruttano formati specifici (**Caligari**, **Sculpt**, **Turbo Silver**, eccetera). Volendo cercare un punto di raccordo con gli Ibm compatibili, l'Iff si presta già come primo punto di contatto, in quanto accettato da alcuni visualizzatori, e sfruttato da qualche tool grafico (**Dpaint2**).

Per la cronaca, i files in questo formato sono di solito caratterizzati dal suffisso "**.LBM**".

Un'immagine IFF 640 x 200 in 32 colori

Un altro formato sfruttabile in entrambe le categorie di computer, può essere il GIF, standard adottato e creato in funzione del network telematico Compuserve, per il quale su Amiga esistono dei visualizzatori (ma soprattutto convertitori) dedicati, mentre la maggiore diffusione è decisamente in ambito Pc.

File rispondenti agli standard appena accennati possono dunque essere grosso modo trasferiti anche senza modifiche tra Amiga e Pc (compatibilmente con i problemi di risoluzione prima riportati), ma in realtà sarà quasi sempre necessario modificare il formato di codifica, almeno se si considera un uso in ambito Ms-Dos.

Gli applicativi di quest'ultimo, infatti, per lo più prediligono altri standard, caratterizzati dai suffissi **PCX** (**Pc Paintbrush**), **IMG** (**Gem** e **Ventura**), **BMP** (**Windows 3**), e vari altri più settoriali, come già si è visto in Amiga.

Nella maggior parte dei casi, dunque, un semplice trasferimento **non** potrà risultare sufficiente, ma occorrerà operare la "traduzione" da uno standard all'altro, che, a seconda delle peculiarità del software disponibile, potrà essere effettuata in ambiente Pc oppure Amiga.

Entrambi i computer, per fortuna, dispongono infatti di programmi adatti allo scopo, e, considerando la facilità di trasporto fisico dei file (grafici o meno che siano) da un sistema all'altro, questa va-

I B M   C O M P A T I B I L I

NOME	Show	scelta risol. pixel	scelta numero colori	NOTE
Convert DP	NO	parziale	NO	Traduce Iff, Pcx, Windows Paint
G.Workshop	SI	scaling	NO	Traduce quasi ogni standard
Hijaak	SI	scheda	NO	Traduce tutti i formati
Cshow	SI	NO	NO	Mostra Gif, Pcx, Pic
Fastgif	SI	NO	NO	Solo visualizzatore Gif
Gifpub	SI	solo VGA	NO	Da Gif a Pcx in toni di grigio
Picem	SI	parz.	parz.	Cambia Gif, Pcx, Img, Pic, Bas

I programmi di "conversione" più diffusi in ambiente Ms-Dos

rietà consente in pratica ogni tipo di manipolazione, se necessario realizzata con più di un "passaggio".

Nei due riquadri di queste pagine sono riassunte le caratteristiche di base di due raggruppamenti di programmi, divisi a seconda della "famiglia" di appartenenza (Amiga e Ms- Dos), che non rappresentano comunque la totalità delle scelte disponibili. Possono però considerarsi quanto di più diffuso tanto nell'ambito del Pubblico Dominio / Shareware che in quello commerciale, e riassumono un po' tutte le possibilità pratiche di cui si dispone per implementare una stessa immagine nei due sistemi.

Vediamoli più da vicino.

I simulatori Ms-Dos per Amiga invitano a sperimentazioni

Su Amiga...

Generalizzando, i programmi mostrati nella tavola riassuntiva possono essere divisi in due categorie: quelli di libera distribuzione e quelli commerciali.

La divisione, nel caso di Amiga, è anche influente sulle prestazioni del software. Non tanto per una distinzione qualitativa, quanto per il fatto che il circuito PD ha finora privilegiato i rapporti tra gli standard grafici Iff e Gif.

Naturalmente, data la loro ampissima diffusione e popolarità, non è stata inclusa nell'elenco la lunga serie di visualizzatori di schermate Iff disponibile, certamente nota a tutti.

Mancano all'appello, come si può constatare, file che mostrino altri formati dell'ambiente Pc, ma anche con quelli disponibili ci si può arrangiare.

I visualizzatori di Gif, il cui uso (obbligatoriamente da ambiente **SHELL**) è estremamente semplice ed illustrato per lo più lanciando i programmi senza parametri, non necessitano di particolari attenzioni. Basta impartire il loro nome seguito da quello del file grafico, e quest'ultimo verrà mostrato a video. C'è da dire che, in genere, risultano tutti piuttosto lenti nella visualizzazione, per cui è più proficuo convertire il file GIF in IFF, e quindi sfruttare i soliti **Show** (e similari).

Come mostrato dalla tabella, alcuni di questi (**Virtgif** e **Hamgif**) sono tra l'altro in grado di "intercettare" l'esistenza di un numero di colori superiori a **32**, per ricorrere di conseguenza alla modalità **Ham** (Hold And Modify) di Amiga, con ottimi risultati (lentezze a parte).

Virtgif, inoltre, presenta una caratteristica peculiare: se la schermata ha una risoluzione in pixel superiore allo schermo in cui lo si è lanciato, opererà la visualizzazione in overscan, ovvero mostrerà solo una porzione dell'immagine, con la possibilità di operare uno scroll per accedere alla rimanente.

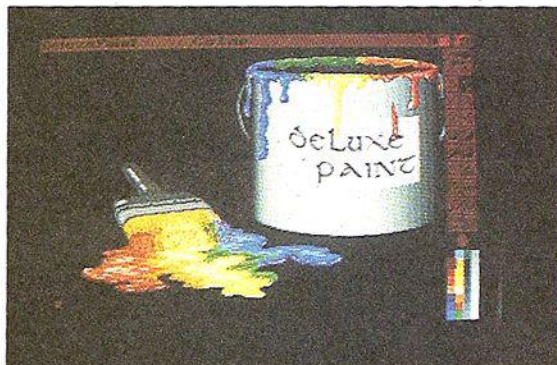
Ma, considerato il tema di queste riflessioni, maggiore importanza assumono certamente gli altri file, quelli in grado di effettuare delle vere **conversioni di formato.** Per ciò che riguarda il Gif, segnaliamo in particolare **HamSharp** e **IffToGif**: il primo converte da Gif a Iff, il secondo svolge il compito inverso. Entrambi leggono un file origine, e trascrivono il nuovo formato su un diverso file di destinazione. Anch'essi operano da Shell, e non richiedono particolari attenzioni nell'uso. Hamsharp, dopo il lancio, chiede esplicitamente che siano forniti, da tastiera, i nomi di input e di output, e, se

Ancora un'immagine HAM 320 x 200 in 4096 colori

necessario, ricorre anch'esso al modo Ham per tradurre in formato Amiga Ilbm (Iff) eventuali immagini con risoluzione colore superiore a 32. Per la conversione opposta, Iff2gif ha invece qualche problema a tradurre il modo Ham in Gif, per il quale comunque esiste sempre Imagelink, come vedremo tra breve.

Tutti questi programmi, si noterà dallo schema in tabella, non prevedono modifiche della risoluzione generale dell'immagine, ovvero la sua dimensione in pixel, né del numero di colori presenti. Varianti, queste, che possono però avere la loro importanza, soprattutto se la traduzione di formato è finalizzata ad un uso dell'immagine in ambiente Pc. Se, indipendentemente dalla (futura) conversione si vuole cambiare la dimensione in pixel dell'immagine, ovviamente senza perdere elementi della schermata, si può comunque ricorrere a **Transfer24**, una utility che si affianca al ben noto **Digipaint**.

Il suo uso è semplicissimo: dopo il lancio, appare una schermata che consente di settare a suon di mouse i parametri di risoluzione dello schermo, ed eventualmente di abolire, o meno, il colore. Ciò fatto, non resta che caricare il file (rigorosamente) Iff desiderato. Quale che sia la sua impostazione grafica, questa verrà tradotta nei nuovi parametri, e mostrata sul monitor. In pratica non è possibile altra modifica, ma si può ovviamente salvare il risultato su file, che sarà anch'esso in formato Iff.

Al top delle prestazioni "migratorie", troviamo comunque **Imagelink**, che non può mancare nella softeca di chiunque smanetti sia con Amiga che con i Pc. Questo programma, dall'impostazione intuition-oriented, propone un tipo di approccio simile a quello delle cosiddette directory utility, con due finestre che mostrano il formato del file di input in una, e quello del file di output nell'altra. Per sceglierlo, è sufficiente cliccare col mouse sulla sua definizione esplicita. La disponibilità è veramente notevole, e può comportare la "traduzione" da un formato Amiga all'altro (per esempio da Iff/Ilbm a Sculpt, a Turbo silver, o il contrario), come pure la trasformazione dei dati grafici in standard "alieni": Gif, Pc e Tiff del mondo Ms - Dos, o addirittura il **Pict di Macintosh**. I formati riferibili ai Pc Ibm compatibili non sono proprio tutti, ma risultano più che sufficienti: se proprio ne giovasse uno diverso, lo si potrebbe "raggiungere" da ambiente Ms - Dos, come vedremo tra breve.

Ma le prestazioni di Imagelink non finiscono qui. Dopo la scelta dello standard di codifica, e quella successiva riguardante i nomi dei file di input e output da trattare (fornita di comodissimo file requester), si accede a due ulteriori schermi dai quali è possibile impostare l'eventuale compressione del file destinazione, la scelta sul numero di colori da adottare (Ham compreso!), e infine le dimensioni in pixel dell'immagine.

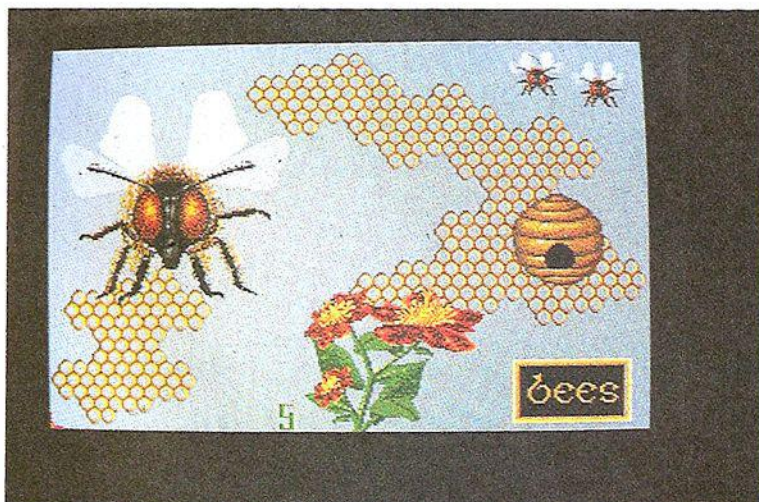
Dimensioni, si badi bene, che comprendono un reale **"resize"** della schermata, che può così essere adattata alle esigenze (per esempio) della scheda grafica che si possiede sul Pc. Il trasferimento, insomma, può essere compiuto con il file già pronto per l'uso, sempre che tra gli standard adoperabili sia compreso quello da noi desiderato.

In caso contrario, non resta che effettuare le eventuali modifiche a carico delle dimensioni e del colore, e passare la palla all'altro computer.

...e su Pc

Le operazioni di traslazione da un formato all'altro, se operate in ambiente Ms - Dos, in un certo senso si fanno più semplici.

Se l'obiettivo è quello del trasferimento di un file verso Amiga, basterà che un qualunque tool di conversione comprenda il formato Iff/Ilbm, o quello Gif, e la cosa risulterà immediatamente fattibile.

Sarà nostra cura, in un secondo momento, adattare eventuali discrepanze di risoluzione adoperando su Amiga uno dei programmi prima descritti. Le utility di

Una schermata Amiga 640 x 256 in 32 colori

ambiente Pc, infatti, non propongono in genere un **"fine tuning"** (aggiustamento fine) a carico della risoluzione in pixel o sul numero di colori, in quanto la scelta è soprattutto rivolta alla presenza di una certa scheda grafica piuttosto che un'altra. Chiaro che, se si dispone per esempio di una Vga avanzata, si avrà per lo più accesso ad una maggiore scelta.

Anche i programmi più diffusi in questo settore sono riassunti nella seconda tabella di queste pagine, che come ovvio **non** possiede la voce Ham.

Come si può notare dispongono tutti di una opzione di visualizzazione dell'immagine, fatta eccezione per il Convert incluso nel software di Deluxe Paint II. Questo è anche tra i pochi file a non possedere una interfaccia utente di uso intuitivo.

Va infatti usato da Dos fornendo il nome del file di input e quello del file di output, unitamente a uno **switch**, una lettera preceduta dal simbolo del meno (-), che specifica il tipo di operazione che deve essere compiuta. Una descrizione degli switch la si ottiene impartendo solo **Convert**, per cui ci limiteremo qui a dire che questo programma consente di trattare i formati **Iff**, **Windows Paint**, e **Pc Paintbrush** (quest'ultimo in varie risoluzioni). Non molto, ma non si dimentichi che, per i nostri scopi, è più che sufficiente.

Tra l'altro Deluxe Paint, tra tutti, è forse l'unico tool che permette di trasferire **da Pc ad Amiga** un file **senza** ulteriori modifiche, in quanto già in formato ".lbm", che poi altri non è che il solito Iff.

Piuttosto sofisticato nelle opzioni, anch'esse però da impartire da Dos unitamente al comando, **Picem** consente qualche interessante variazione sul tema, pur se dedicato principalmente alla visualizzazione di file grafici.

Intanto, i formati "supportati" sono parecchi, e comprendono **Gif**, **Pcx**, **Img**, e vari altri di minore interesse per i nostri scopi. Sorvolando sulle opzioni, ampiamente descritte dall'help iniziale del programma (3 schermate!), ciò che può interessare è la capacità di modificare, dopo la visualizzazione di un file grafico, parametri come il contrasto generale dell'immagine, o la sua dimensione rispetto allo schermo.

Feature, queste, che diventano importanti se si considera che la stessa immagine può poi essere salvata con le caratteristiche acquisite, pronta eventualmente (in Gif, l'Iff **non** è supportato) per ulteriori ritocchi o per il salto verso l'universo Amiga.

Nella tabella, come già per Amiga, sono riportati anche due file dediti unicamente alla visualizzazione, **Fastgif** e **Cshow**, con quest'ultimo dotato di interfaccia semigrafica in grado di consentire la selezione dei file (che possono essere in formato Gif, Pcx oppure Pic) direttamente da schermo, spostandosi di directory in directory col semplice movimento dei tasti cursore.

Molto ampia, inoltre, la scelta della risoluzione video, che può andare dalla vecchia **Cga** fino alle più moderne **Super Vga**. Il formato originale del file da visualizzare viene tra l'altro precisato in tutte le sue componenti, ivi compreso il numero di colori, in modo da facilitare l'impostazione di una specifica risoluzione.

La stessa facilità di approccio, ma con in più delle vere possibilità di conversione, la si può riscontrare nei due programmi più significativi del settore, ovvero **Graphic Workshop** e **Hijaak**.

Pur se appartenente al settore dello Shareware, il primo è di qualità decisamente notevole. In pratica, attraverso comodi spostamenti di directory realizzate attraverso l'uso dei tasti cursore, una volta selezionato un file di pressoché qualunque standard (MacPaint, Gem/Ventura, Pcx, Gif, Iff, Tiff, Wordperfect, Windows3, Pic) compreso il **Postscript**(!), questo viene mostrato a video, eventualmente in overscan se la dimensione eccede quella del video (è scheda grafica permettendo).

Ma le risorse più interessanti sono raggiungibili tramite i tasti funzione del Pc, la cui interpretazione è chiaramente riportata sullo schermo: **rotazione** dell'immagine, **rescaling** (modifica della dimensione orizzontale, verticale, oppure entrambe), **stampa** su carta, **inversione**, variazione del **contrasto**, e naturalmente **salvataggio** in uno standard diverso da quello originario. Visto che l'Iff e il Gif sono implementati, trasferire poi il file su Amiga e lì utilizzarlo direttamente, risulta uno scherzo.

Ultimo, ma non certo per demerito, va poi citato **Hijaak**, un software commerciale che forse è **il più valido e flessibile software** tra quelli finora descritti. Anch'esso dotato di interfaccia intuitiva, dispone di 3 menu principali, attraverso i quali settare le varie preferenze, fornire in input i dati del file da convertire e quelli del file destinazione, oppure installare un modo **capture**. Questo, in pratica, consentirà in qualunque momento di catturare (guarda caso) il contenuto dello schermo in un file grafico.

La dotazione di **standard** è enorme: ben **29**, che comprendono tutti quelli finora citati e alcuni molto particolari come i **Fax compatibili**, o il Postscript. Per la scelta, vengono mostrati direttamente a video, ove li si può facilmente far scrollare per poi selezionarli con il tasto Return. Va detto, per la goduria dei fan commodoriani, che questo è l'unico programma che definisce a chiare lettere un formato **Amiga Ilbm**, piuttosto che il solito "anonimo" Iff. Il suo uso generale risulta sufficientemente semplice, ma per sfruttarlo realmente al massimo richiede una consultazione del manuale.

Come si è potuto constatare in queste (necessariamente) brevi note, la proverbiale incompatibilità tra due computer di classi così diverse non è poi così totale. La grafica, almeno quella più facilmente manipolabile in entrambi gli ambienti, può essere senza dubbio un motivo di avvicinamento, piuttosto che di separazione.

Il resto, come ovvio, dipende dagli incostanti umori di una utenza che dire eterogenea è forse poco...

Francesco Capuzzi

Escape, illustre sconosciuto

Quattro chiacchiere su come inviare comandi di Escape al "sistema" Amiga

Quante volte, nell'attesa del caricamento del WorkBench, vi sarà venuta la tentazione di interrompere l'esecuzione della Startup-Sequence, cioè di quella serie di comandi che svolge il computer quando si inserisce il dischetto: impossibile, diranno in molti. Ma con Amiga tutto è possibile, basta saper come fare. Ebbene, forse non tutti conoscono gli utili comandi attivabili sfruttando il tasto **Ctrl** riportati in tabella, che possono essere attivati digitando a casaccio qualcosa sullo schermo e sperimentando a volontà.

Per restare sempre in argomento, ora parleremo di come si possa cambiare lo stile ed il colore dei caratteri, per mezzo del comando **Echo**. Chi possiede una stampante, avrà quasi sicuramente sentito parlare delle sequenze di **Escape**: sono sequenze di caratteri precedute, appunto, dal codice Escape, il famoso **Chr$ (27)**, utilizzate per definire la spaziatura, lo stile, la tabulazione.

Visto che **Cli** (oppure Shell) non accetta l'immissione del carattere di Escape, che *sembrerebbe* possibile inserire con la semplice pressione del **tasto Esc** (in alto a sinistra di Amiga), chi ha programmato il comando Echo ha pensato anche a questo: se si vuole inviare il codice Escape, si dovrà scrivere Echo ***asterisco e*** (***e**), dove la coppia di caratteri ***e** comunica, appunto, che si tratta del codice Escape. Escape dovrà quindi essere simulato da un gruppo di caratteri, a seconda di ciò che si vuol fare. Come esempio si può prendere visione della tabella riportata in queste pagine, tenendo presente che i caratteri che rappresentano i comandi (compreso il codice di Escape) vanno racchiusi tra virgolette; cioè, ad esempio: **"*e[3m"** e non semplicemente ***e[3m**.

Inoltre è possibile cambiare il colore di primo piano dei caratteri con **Esc [3nm**, dove **n** è un qualunque numero da 0 a 8 (nel Cli varranno, però, solo i colori da 0 a 2), mentre per cambiare il colore di sottofondo si dovrà scrivere **Esc [4nm**, dove **n** è sempre un qualunque numero da 0 a 8. E' bene ricordare che nella tabella pubblicata il simbolo Esc rappresenta solo un modo di indicare Escape; volendo, per esempio, scrivere in corsivo, si dovrà digitare

```
Echo "*e[3m"
```

...e non...

```
Echo "Esc [3m"
```

Le sequenze appena viste sono dette anche ANSI compatibili, cioè vengono comprese da qualunque protocollo di scambio dati che utilizzi lo standard ANSI (per esempio lo scambio dati via modem). E' inoltre possibile compiere direttamente le operazioni descritte, senza utilizzare il comando Echo: ad esempio, volendo scrivere in nero, si dovranno premere i tasti **Esc**, **[**, **2** e **m** (premendo infine **Return**), azione che può generare un messaggio di errore, ma esegue comunque il comando impartito.

Vi starete già chiedendo: a che serve allora utilizzare il comando Echo? Presto detto: nei file batch, cioè in quei file contenenti comandi Dos. Ad esempio, volendo scrivere in **grassetto**, sottolineato, arancione su fondo bianco, si dovrà ogni volta digitare...

```
Echo "*e[1m*e[4m*e[33m*e[31m"
```

...oppure impartire direttamente le sequenze via tastiera (guai, però, a sbagliare la digitazione!). Se però si vuole che, durante il caricamento della startup-sequence (il file batch che il sistema esegue all'inserimento del primo dischetto) appaia un messaggio con diversi stili di scrittura, non sarà possibile impartire le sequenze da tastiera, e si dovrà per forza far ricorso al comando Echo.

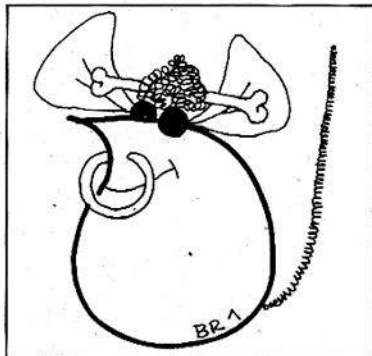
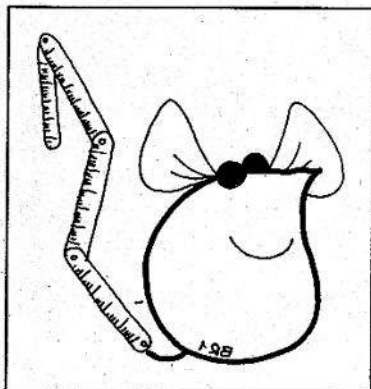
Il programma

Il listato proposto (che è un file Batch) fa proprio questo: dopo aver pulito lo schermo, visualizza i vari stili di scrittura possibili e i vari colori.

Per provarlo, attivate un qualunque editor, digitate il listato pubblicato e registratelo assegnando un nome qualsiasi, ad esempio **ProvaCli**.

In seguito, dopo aver caricato il Cli (oppure Shell), impartite...

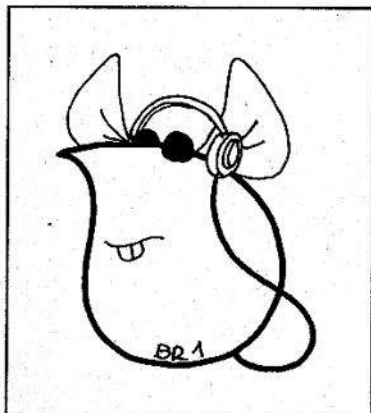
```
Execute ProvaCli
```

...oppure, invece di *ProvaCli*, in nome del file con cui si era precedentemente registrato il listato.

Le principali sequenze di Escape attivabili in Amiga

Esc c = Riporta allo stato iniziale
Esc D = Scende di una linea, senza però andare a capo
Esc E = Va a capo
Esc M = Sale di una linea
Esc [0m = Riporta lo stile e il colore allo stato iniziale
Esc [3m = Stile corsivo
Esc [4m = Stile sottolineato
Esc [1m = Stile grassetto
Esc [2m = Scrive in nero (Equivale a Esc [32m)
Esc [7m = Inverte il colore di primo piano con quello di fondo
Esc [8m = Blocca l'emissione di testi

```
;* CLI Look              *
;* by Francesco Capuzzi *
;* Carbonia (CA)        *
   echo "*ec"
   echo "*e[4m*e[32m CLI Look "
   echo "*e[0m"
   echo "*e[1m     Boldface (Grassetto)*e[0m"
   echo "*e[3m     Italic   (Corsivo  )"
   echo "*e[4m     Corsivo sottolineato "
   echo "*e[1m     Corsivo grassetto sottolineato"
   echo "*eE*e[0m"
   echo "*e[33m    Arancione e *e[7mReverse*e[0m"
   echo "   *e[2m*e[41mNero su bianco"
   echo "*e[8m Questa frase non apparira'. "
   echo "*e[0m Scrittura normale."
```

Il file Batch di cui si parla nell'articolo

E' bene ricordare, soprattutto ai principianti (cui dedichiamo queste pagine) che il programma riportato a fianco deve esser digitato con un Text editor. Si consiglia, in alternativa, di usare un programma di videoscrittura purchè questo non inserisca, oltre ai caratteri alfanumerici, i codici di controllo tipici di un word processor. Usate, quindi, l'opzione ASCII al momento del salvataggio del file; tale opzione è sempre disponibile nei moderni programmi di w/p.

Ctrl + **D** = Ferma l'esecuzione della startup-sequence
Ctrl + **L** = Cancella il contenuto della finestra del Cli
Ctrl + **X** = Cancella l'intera riga corrente
Ctrl + **N** = Imposta il set caratteri alternativo
Ctrl + **O** = Disattiva il set caratteri alternativo
Ctrl + **J** = Va a capo
Ctrl + **K** = Muove il cursore in alto
Ctrl + **I** = Equivale al tasto TAB
Ctrl + **M** = Equivale a Return (come nel C/64!)
Ctrl + **G** = Lampeggio momentaneo dello schermo
Ctrl + **H** = Cancella un carattere
Ctrl + \ = Invia il comando senza andare a capo

I comandi eseguibili, direttamente da Shell, sfruttando Ctrl

di Francesco Capuzzi

Amiga, super routine assembler per Hard Copy

Se vi piacciono gli argomenti "tosti", e desiderate applicazioni pratiche, questa routine di impiego universale sarà pane per i vostri denti

Parte della fama di Amiga è dovuta soprattutto alla sua potenza di programmazione.

Infatti una buona parte della sua Rom (oltre **192 K**) è occupata da routine di semplice utilizzo che, rendendo meno dura la vita del programmatore, possono essere raggruppate in librerie o in device.

Le **librerie** possono risiedere in memoria, come *Exec, Intuition, Graphics* e *Math*, o nella directory **Libs** del disco di startup, cioè quello che si è inserito dopo l'accensione di Amiga.

Lo stesso discorso può essere fatto per i **device**. Ne esistono alcuni (come i device *Input, Keyboard, Trackdisk*) che risiedono in memoria; altri (come i device *Printer, Parallel, Narrator*), invece, risiedono nella directory **Devs** del disco di startup.

A differenza delle librerie, i device non sono di immediato utilizzo, ma prima di accedervi devono essere svolte alcune operazioni con l'ausilio delle funzioni della libreria di **Exec**. Infatti un'altra caratteristica dei device è la loro **dipendenza dalle librerie** dal momento che possono essere aperti solo attraverso le funzioni di Exec a loro dedicate. Le operazioni da compiere per aprire un device non sono complesse, ma necessitano di una certa conoscenza del sistema operativo di Amiga.

La libreria di Exec utilizza, per lo scambio di dati, alcune strutture che sono alla base del sistema operativo.

in questo articolo ci occuperemo di quelle riguardanti l'apertura e la gestione dei device, e in particolare del device Printer.

Strutture

Una **struttura**, per chi non lo sapesse, è un modo convenzionale di suddividere una parte di memoria.

In altre parole si tratta di routine Im contenenti gruppi di byte individuabili mediante nomi convenzionali (riconosciuti dai compilatori Assembler più diffusi) che devono esser settati in modo opportuno prima di attivare la routine Im cui appartengono.

La routine, ovviamente, gestirà quindi i valori "passati" in modo totalmente automatico e trasparente per l'utente.

La struttura più importante è la **IORequest** (**I**nput/**O**utput **Request** = Richiesta di input/output), da passare a Exec all'apertura del device:

```
STRUCTURE   IO,MN_SIZE
   APTR     IO_DEVICE
   APTR     IO_UNIT
```

```
UWORD    IO_COMMAND
UBYTE    IO_FLAGS
BYTE     IO_ERROR
```

La struttura non è ugualmente idonea per tutti i device.

L'unico parametro da inizializzare (gli altri devono essere lasciati a zero) è **Io_command**, che deve contenere il **numero** del comando da comunicare al device, di cui parleremo in seguito.

La prima linea della struttura, cioè **Structure io, mn_size**, significa che questa deve essere preceduta dalla struttura formata dalle iniziali **MN**(Message Node):

```
STRUCTURE MN,LN_SIZE
    APTR   MN_REPLYPORT
    UWORD  MN_LENGHT
```

Il parametro **Mn_replyport** deve contenere l'indirizzo di una porta messaggio (descritta più avanti).

Prima di questa struttura ne è presente un'altra, dalle iniziali **LN** (List Node = Nodo di Lista):

```
STRUCTURE LN,0
    APTR   LN_SUCC
    APTR   LN_PRED
    UBYTE  LN_TYPE
    BYTE   LN_PRI
    APTR   LN_NAME
```

Il parametro **Ln_type** deve contenere il valore **5**, per avvertire Exec che è il nodo di lista di un messaggio.

Tale struttura è infatti utilizzata da Exec per molti scopi: nelle strutture Task, interrupt, device, port e in molte altre.

Quindi il parametro **Type** è di vitale importanza per far capire a Exec lo scopo per cui è utilizzata.

Il parametro **Ln_pri**, contenente un valore compreso fra **-128** e **127**, indica la priorità del nodo.

Il parametro **Ln_name** deve contenere un puntatore alla stringa Null-terminated del nome della Lista, che può essere uno qualunque.

Il parametro **Mn_reply Port**, come prima affermato, deve contenere l'indirizzo di una porta messaggio. Anche questa, essendo una struttura Port, comprende la struttura List Node:

```
STRUCTURE MP,LN_SIZE
  UBYTE   MP_FLAGS
  UBYTE   MP_SIGBIT
  APTR    MP_SIGTASK
  STRUCT  MP_MSGLIST,LH_SIZE
```

Stavolta il valore di **Ln_type** deve essere **4** (invece di 5), perché la struttura è utilizzata per definire una message Port.

Per spiegare il significato dei parametri Flags, Sigbit e Sigtask è però necessario conoscere il meccanismo con cui Exec e i task si scambiano messaggi.

Ogni task, cioè un programma che gira indipendentemente dagli altri, può essere in due stati: **attivo** o **in attesa** di un messaggio. Può essere ad esempio in attesa di un messaggio proveniente da un altro task, che deve arrivare a una message port.

Vi sono due modi per attendere un messaggio: utilizzando la funzione **WaitPort(Port)**, in cui *Port* è il puntatore alla porta in cui dovrebbe arrivare il messaggio; oppure utilizzando la funzione **Wait(SigSet)**, che può attendere più messaggi contemporaneamente. Ogni task, infatti, ha a disposizione **trentadue bit** per altrettanti segnali: i primi sedici sono utilizzati internamente da Exec, mentre i segnali da sedici a trentuno possono essere abbinati a una porta.

Il flag *Sigbit* serve proprio a definire quale segnale è abbinato a questa porta; un segnale può essere allocato con la funzione **AllocSignal(Signal)**, dove *Signal* deve contenere o il numero del segnale che si vuole allocare (solo da 16 a 31) oppure **-1**, valore che alloca un segnale qualunque, il cui numero è restituito dalla funzione.

Il parametro *Flags* serve a definire la funzione da svolgere quando un messaggio arriva alla porta: se ha valore **0**, viene segnalato; se ha valore **1** si ha un interrupt software (argomento che esula dagli scopi del presente articolo); se ha valore **2**, il messaggio viene ignorato e accodato alla lista di messaggi che attendono di essere letti.

Finché, infatti, non vengono letti (e quindi rimossi), i messaggi si accodano nella lista della porta. Il parametro *Sigtask* indica il task a cui appartiene la porta. La struttura **LH**(List Header = intestazione di lista), che segue la struttura MessagePort, è il "luogo" in cui vengono accumulati i segnali non rimossi.

```
STRUCTURE LH,0
    APTR   LH_HEAD
    APTR   LH_TAIL
    APTR   LH_TAILPRED
    UBYTE  LH_TYPE
    UBYTE  LH_PAD
```

Tutti i parametri devono essere lasciati a zero, perché vengono inizializzati correttamente da Exec alla chiamata di OpenDevice.

Ma torniamo a spiegare la differenza fra *WaitPort* e *Wait* con un esempio.

Se un task non può stabilire se arriverà prima un messaggio proveniente da un generico **task 1** (nella porta a cui è assegnato il segnale **17**), o un messaggio proveniente da un **task 2** (nella porta a

cui è assegnato il segnale **18**), non può porsi in attesa di un ipotetico messaggio nella porta 17: potrebbe infatti giungere, per primo, un segnale sulla porta 18.

L'unica funzione che si può utilizzare è quindi **Wait(SigSet)**, in cui *SigSet* è una long word che indica in quali porte si attende il messaggio.

Ad esempio, se SigSet fosse **$00000001**, significherebbe che si attende un messaggio destinato alla porta 1 (il bit 1 è On).

Si deve però esser certi dell'arrivo imminente di un messaggio: nel caso non arrivasse, altrimenti, il task rimarrà in un eterno stato di attesa.

Dopo aver esaminato la struttura per l'apertura di un device, vediamo in che consiste il parametro *Command* della struttura *IORequest*.

Esistono alcuni comandi standard, accettati da quasi tutti i device, che sono:

```
CMD_INVALID  (0)
CMD_RESET    (1)
CMD_READ     (2)
CMD_WRITE    (3)
CMD_UPDATE   (4)
CMD_CLEAR    (5)
CMD_STOP     (6)
CMD_START    (7)
CMD_FLUSH    (8)
```

Il numero tra parentesi rappresenta il numero del comando da inserire nel parametro Command. *Cmd_Invalid* invia un comando non valido; *Cmd_Reset* riporta il device allo stato iniziale (quando è stato aperto); *Cmd_Read* legge dati dal device (ovviamente non sarà valido per tutti i device, visto che dalla stampante non si può leggere niente...); *Cmd_Write* invia dati al device; *Cmd_Update* aspetta che l'intero contenuto del buffer (se il device ne ha uno) sia svuotato prima di continuare.

Il buffer viene usato dai device per immagazzinarvi dati, prima che possano essere utilizzati.

Una stampante non può infatti stampare alla stessa velocità con cui arrivano i dati, per cui deve immagazzinarli temporaneamente.

Cmd_Clear svuota l'intero contenuto del buffer; *Cmd_Stop* interrompe momentaneamente l'operazione in corso finché non si invia il comando *Cmd_Start*, che fa riprendere l'esecuzione; *Cmd_Flush* svuota irrimediabilmente il buffer e interrompe, senza possibilità di essere ripresa, qualunque operazione in corso.

Oltre ai comandi già visti, ogni device ha comandi speciali; i comandi del device printer sono tre:

```
PRD_RAWWRITE      (9)
PRD_PRTCOMMAND    (10)
PRD_DUMPRPORT     (11)
```

Prd_Rawwrite invia dati alla stampante; *Prd_Prtcommand* invia invece i comandi.

Prd_Dumprport, l'unico comando di cui ci occuperemo, serve proprio a ottenere, su carta, la copia della RastPort che gli comunichiamo.

La **RastPort** è una struttura, utilizzata dalla libraria Graphics, per memorizzare dati riguardanti la posizione del pennello grafico, il pattern, il tipo di scrittura, il colore corrente etc. ed è propria di ogni finestra, rispettando anche i colori e le dimensioni imposte.

Per comunicare al device le dimensioni di stampa, la Rast Port etc., esiste una struttura apposita, chiamata **IODRPReq** (Input/Output Dump RastPort Request= richiesta di input/output per il trasferimento della Rast Port), che deve obbligatoriamente trovarsi subito dopo la struttura IORequest, come specificato dalla prima riga della struttura stessa:

```
STRUCTURE IODRPReq,IO_SIZE
   APTR    io_RastPort
   APTR    io_ColorMap
   ULONG   io_Modes
   UWORD   io_SrcX
   UWORD   io_SrcY
   UWORD   io_SrcWidth
   UWORD   io_SrcHeight
   ULONG   io_DestCols
   ULONG   io_DestRows
   UWORD   io_Special
```

Il parametro *io_RastPort* deve contenere il puntatore alla RastPort della finestra; *Io_ColorMap* deve contenere l'indirizzo della ColorMap dello schermo in cui è visualizzata la finestra, che è una struttura che contiene le percentuali di rosso, verde e blu dei colori utilizzati nella finestra; il parametro *Modes* deve contenere il modo di visualizzazione (View Mode) della ViewPort dello Schermo in cui si trova la finestra (la ViewPort è la struttura che contiene i dati relativi alla larghezza, all'altezza, ai colori etc. di uno schermo).

Gli altri parametri sono spiegati nel listato in Assembler. Per chi non dispone della documentazione ufficiale o di file include, riportiamo in tabella gli indirizzi delle strutture richieste, a patto però di conoscere l'indirizzo della finestra o dello screen.

Prima abbiamo parlato di **View Mode** che, lo ricordiamo, è il modo in cui viene visualizzato lo schermo; come tutti sanno, uno schermo può essere in bassa, in alta risoluzione, interlacciato o meno, in HAM (Hold And Modify = Tieni e modifica) etc.:

```
V_HIRES equ   $8000   ; Alta risoluz.
V_LAC   equ   4       ; Interlacciato
V_HAM   equ   $800    ; 4096 colori
V_SPRITES equ  $4000  ; Può contenere sprites
```

Resta ancora da spiegare che cosa hanno da spartire Porte e Messaggi con i device.

Dopo che si è aperto il device, si deve inviare il comando vero e proprio al device, operazione che può essere svolta in due modi: per mezzo della funzione **DoIO**, oppure delle funzioni **SendIO, CheckIO, AbortIO** e **WaitIO**.

La funzione *DoIO* è totalmente automatica perché invia un messaggio al device, mettendosi in stato di attesa, inviandogli la struttura IoRequest, dove, come già visto, è presente l'indirizzo di una porta creata da noi.

Quando il device ha completato la richiesta o si è verificato un errore, restituisce il messaggio proprio a questa porta.

Gli indirizzi delle strutture

```
Rast Port = Indirizzo dello schermo + 84
            Peekl (indirizzo della finestra + 50)
View Port = Indirizzo dello schermo + 44
Color Map = Peekl (indirizzo della View Port + 4)
View Mode = Peekw (indirizzo della View Port + 32)
Schermo   = Peekl (indirizzo della finestra + 46)
```

Quando il messaggio vi giunge, il programma riprende la normale esecuzione; tutte le operazioni descritte sono svolte automaticamente dalla funzione DoIO. Ovviamente, se si utilizza DoIO, non sarà necessario utilizzare SendIO o WaitIO.

Il compito della funzione *SendIO* è solo quello di inviare il messaggio, senza aspettare che sia restituito; per questo, l'esecuzione del programma riprende subito.

Se, dopo aver richiamato la funzione SendIO, si vuole controllare che tutto proceda bene, si utilizza la funzione *CheckIO*; se si vuole aspettare finché il device non restituisce il messaggio, e cioè finché non ha finito, si utilizzerà la funzione *WaitIO*.

Infine, se si vuole interrompere la funzione in corso di svolgimento da parte del device, dopo che si era richiamata la funzione SendIO, si può utilizzare la funzione *AbortIO*.

Un eventuale errore verrà segnalato nel parametro **Error** della struttura IORequest, con i seguenti valori a seconda dell'errore:

```
CANCEL           = 1
NOTGRAPHICS      = 2
INVERTHAM        = 3
BADDIMENSION     = 4
DIMENSIONOVFLOW  = 5
INTERNALMEMORY   = 6
BUFFERMEMORY     = 7
```

Cancel segnala che è stata bloccata la fase di stampa; *Notgraphics* segnala che la stampante selezionata per mezzo di Preferences non è una stampante grafica, e quindi non può stampare disegni; *Invertham* segnala che non è possibile stampare schermate in HAM; *Baddimension* e *Dimensionovflow* segnalano che le dimensioni sono errate; gli ultimi due, di solito, segnalano che non è disponibile una quantità sufficiente di memoria.

I listati

Sui due listati, uno in **Assembler**, l'altro in **Basic**, non c'è molto da dire.

Quello in Assembler segue la sintassi **DevPac**.

Dopo aver assemblato il programma selezionando l'opzione *Debug* (dai menu dell'editor) si deve premere **S**: verrà chiesto con quale nome registrare il programma; digitate `df0:HCopy` e premete Return.

A questo punto saranno chiesti gli estremi del programma, cioè l'indirizzo di inizio e di fine dello stesso; scrivete, appunto, Inizio, Fine (vedi listato Assembler) e premete ancora Return.

A questo punto si può caricare il listato in Basic che è solo una dimostrazione del funzionamento della routine proposta; dapprima carica il file precedentemente compilato e registrato, che contiene la routine in LM, poi apre uno schermo a trentadue colori e vi disegna una tavolozza.

Tenendo premuto il pulsante **sinistro**, e muovendo il **mouse**, potrete cambiare le dimensioni di stampa della tavolozza, prestando però attenzione al fatto che sono espresse in millesimi di pollice (un pollice equivale a 2.54 centimetri).

Scelte le dimensioni, avrete tre possibilità: premendo **Q** il programma terminerà.

Con **P** stamperà la tavolozza rispettando le dimensioni imposte; premendo **N** la tavolozza verrà stampata rispettando il rapporto fra i due lati del rettangolo da stampare secondo la proporzione...

X : Y = XS : YS

...in cui X ed Y sono i lati del rettangolo da stampare, e XS e YS le dimensioni di stampa (per esempio, se X = 100, Y = 50, XS = 1000 e YS = 2000, YS sarà trasformato in 500).

Alla funzione si devono passare i seguenti parametri...

```
CALL HCopy& (Finestra&, X&,
Y&, X1&, Y1&, XS&, YS&, F&)
```

...il cui significato è presto chiarito:

Finestra = Indirizzo della finestra (ottenibile in Basic con la funzione Window(7))

X&, Y& = Coordinate angolo superiore sinistro

X1&, Y1&= Dimensioni del rettangolo da stampare

XS&, YS&= dimensioni di stampa

F& = Possibili opzioni:

- **1**= La stampa avverrà nel centro del foglio
- **2**= X1&:Y1&=XS&:YS& (vedi sopra)
- **3**= Opzioni 1 e 2

L'unica imposizione presentata della routine è relativa alla tassativa presenza, nella directory **Devs** del disco di startup, del file **Printer.device**.

In caso contrario si potrebbero generare errori difficilmente individuabili.

```
REM ** Demo di Hard Copy **
REM ** Scritto da        **
REM ** Capuzzi Francesco **
REM **     Carbonia (CA)  **

  OPEN "df0:HCopy" FOR INPUT AS 1
   prog$=INPUT$(LOF(1),1)
  CLOSE 1
1 HCopy&=SADD(prog$)
IF HCopy&/2<> INT(HCopy&/2) THEN prog$=prog$+"":GOTO 1
  SCREEN 1,172,115,5,1
  WINDOW 2," Hard Copy",(0,0)-(163,100),0,1
  COLOR 2,1
  CLS
  PRINT  "Prova di stampa : "
   FOR f=0 TO 1
    FOR c=0 TO 15
     LINE (2+c*10,15+y)-(2+c*10+10,25+y),c+d,bf
    NEXT c
    y=y+10
    d=d+16
  NEXT f
  LINE (2,15)-(160,35),2,b
Scrive:
  LOCATE 9,1
  PRINT  " X = "x&;:PRINT " Y ="y&;"    ":PRINT
  PRINT " Premi <P> <N> <Q>
  f&=1
```

```
Loop:
   a$=INKEY$:a=MOUSE(0):IF a$="" AND a=0 THEN
Loop
     IF a THEN
      x&=MOUSE(1)*100
      y&=MOUSE(2)*100
       IF x&>8000 THEN x&=8000
       IF y&>10000 THEN y&=10000
      GOTO Scrive:
     END IF
   IF a$="q" THEN Fine
     IF a$="p" THEN Stampa
     IF a$="n" THEN f&=3:GOTO Stampa
     GOTO Loop
Stampa:
   a&=WINDOW(7)
   CALL HCopy&(a&,0&,0&,161&,37&,x&,y&,f&)
   BEEP
   GOTO Loop
Fine:
   WINDOW CLOSE 2
   SCREEN CLOSE 1
```

La seconda parte del listato Basic

Il listato (versione DevPac) che genera la parte in lm

```
              opt p+
**  Hard Copy II          **
** by Francesco Capuzzi   **
**       Carbonia (CA)    **

_SysBase         equ $4
_LVOFindTask     equ -294
_LVOAllocSignal  equ -330
_LVOAddPort      equ -354
_LVOOpenDevice   equ -444
_LVOCloseDevice  equ -450
_LVORemPort      equ -360
_LVOFreeSignal   equ -336
_LVODoIO         equ -456
CALLEXEC MACRO                     ; Macro che richiama le funzioni
          move.l _SysBase,a6       ; di EXEC
          jsr _LVO\1(a6)
         ENDM
Inizio    movem.l d0-d2/a0-a2,-(sp) ; Salva i registri nello stack
          move.l 28(sp),a0          ; Preleva l'indirizzo della finestra
          lea.l IODRPReq(pc),a1     ; Mette l'indirizzo della struttura in A1
          move.l $32(a0),(a1)+      ; Ci scrive l'indirizzo della Rast Port
          move.l $2E(a0),a0         ; Prende l'indirizzo dello schermo
          move.l $30(a0),(a1)+      ; e l'indirizzo della colormap
          addq #2,a1                ; Per compatibilita' di formato
          move.w $4C(a0),(a1)+      ; Prende il modo di visualizzazione
          move.w 34(sp),(a1)+       ; Prende dallo stack le coordinate,
          move.w 38(sp),(a1)+       ;   "      "      "      "      "
          move.w 42(sp),(a1)+       ; le dimensioni della figura
          move.w 46(sp),(a1)+       ;   "      "      "      "      "
          move.l 48(sp),(a1)+       ; e le dimensioni di stampa.
          move.l 52(sp),(a1)+       ;   "      "      "      "      "
          btst.b #0,59(sp)          ; Il bit 0 del parametro flag e' 1 ?
          beq Cont                  ; se no , va alla label cont
          ori.w #$40,(a1)           ; si : setta il bit di centratura (BIT 6)
Cont      btst.b #1,59(sp)          ; Il bit 1 del parametro flag e' 1 ?
          beq Cont1                 ; se no , va alla label Cont1
          ori.w #$80,(a1)           ; si : setta il bit di aspetto (BIT 7)
Cont1     move.l #0,a1              ; Richiede a EXEC qual e' il task
          CALLEXEC FindTask         ; correntemente in uso
          lea.l _Task(pc),a0        ; Ne prende l'indirizzo
          move.l d0,(a0)            ; e lo scrive nella struttura Port
          moveq #-1,d0              ; Alloca un segnale qualunque
          CALLEXEC AllocSignal      ; da utilizzare nella porta messaggio
```

```
         lea.l _SigBit(pc),a0        ; e lo scrive nella struttura
         move.b d0,(a0)              ; Port
         lea.l Port(pc),a1           ; Linka questa struttura alla struttura
         lea.l _MSGPort(pc),a0       ; di richiesta di input/output
         move.l a1,(a0)              ; scrivendone l'indirizzo
         move.l PortName(pc),10(a1)  ; e il nome ( a piacere ),e aggiunge
         CALLEXEC AddPort            ; questa porta alla lista di sistema
         lea.l IORequest(pc),a1      ; Passa l'indirizzo della struttura di
         lea.l DevName(pc),a0        ; richiesta di input/output alla funzione
         moveq #0,d0                 ; Nessuna Unita' specifica
         moveq #0,d1                 ; Nessun flag
         CALLEXEC OpenDevice         ; Apre il device
         tst.l d0                    ; C'e' stato un errore ?
         bne Err                     ; se d0<>0 si.
         lea.l IORequest(pc),a1      ; Passa alla funzione l'indirizzo della
         CALLEXEC DoIO               ; struttura e la invia alla stampante .
         lea.l IORequest(pc),a1      ; Chiude il device.
         CALLEXEC CloseDevice        ;
Err      lea.l Port(pc),a1           ; Rimuove la porta
         CALLEXEC RemPort            ;
         move.b _SigBit(pc),d0       ; Restituisce il segnale
         CALLEXEC FreeSignal         ;
         movem.l (sp)+,d0-d2/a0-a2   ; Ripristina il valore dei registri
         RTS                         ; e ritorna al Basic
Port
         dc.l 0,0
         dc.b 4          ; Tipo di nodo (4=MSGPort)
         dc.b 0
         dc.l 0
         dc.b 0
_SigBit  dc.b 0          ; Segnale allocato
_Task    dc.l 0          ; Task
         dc.l 0
         dc.l 0,0
         dc.w 0
IORequest
         dc.l 0,0
         dc.b 5          ; Tipo di nodo (5=Message)
         dc.b 0
         dc.l 0
_MSGPort dc.l 0          ; Porta di attesa
         dc.w 0
         dc.l 0
         dc.l 0
Command  dc.w 11         ; Comando
         dc.b 0,0
IODRPReq
         dc.l 0  ; Indirizzo della RastPort
         dc.l 0  ;       "       "   ColorMap
         dc.l 0  ; Modo di visualizzazione
         dc.w 0  ; Coordinata X di inizio
         dc.w 0  ;    "       Y    "
         dc.w 0  ; Larghezza del rettangolo
         dc.w 0  ; Altezza del rettangolo
         dc.l 0  ; Dimensioni della destinazione
         dc.l 0  ;      "        "        "
         dc.w $3 ; modo di stampa

PortName dc.b "Porta parallela",0 ; Qualunque nome va bene
         cnop 0,2
DevName  dc.b "printer.device",0  ; Nome del device
         cnop 0,2
Fine
```

di Alessandro Andreuccetti

Un archivio in AmigaC per ordinare i nostri dischi

Ovvero: come tenere in ordine i propri files con poca fatica

```
echo;/* Queste righe occorrono per la compilazione automatica del file
lc -vbr filcat.c
blink FROM lib:c.o+filcat.o TO filcat SC SD ND VERBOSE LIBRARY
lib:lc.lib+lib:amiga.lib
quit
*/
/* filcat.c --- V1.0 --- by Alessandro Andreuccetti
**
** Legge un disco con il comando 'Dir opt a' e ne reindirizza l'output
** su un file temporaneo che a sua volta viene letto dal programma e
** vagliato. Eventualmente e' possibile scegliere i files da salvare
** tramite una opzione che consente operazioni interattive.
** Per compilare e linkare questo file sorgente e' sufficiente digitare
** da Shell il comando 'Execute filcat.c' come se fosse un file script
** (questa tecnica e' ripresa pari pari da una sorgente di John Toebes
** ed e' utile per la compilazione di singoli files)
**
** Last Change: 2-mar-91
*/
#include <exec/types.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stddef.h>
#include <libraries/dos.h>
#include <libraries/dosextens.h>

/* flags di stato */
#define INTER 1
#define DEFAULT 0
SHORT mode=0;

/* variabili esterne */
BYTE  disk[80];
FILE *stream;
/*--------------------------------------
** scrive una stringa usando la funzione
** AmigaDOS Write()
*/
VOID PutString(string)
BYTE *string;
{
Write(Output(),string,strlen(string));
```

Il listato in Amiga C: prima parte

Uno dei primi problemi che, inevitabilmente, incontrano tutti coloro che passano parte del loro tempo sulla tastiera del loro amico elettronico è, di sicuro, quello della ricerca di files e programmi, spesso vitali per la nostra attività smanettona, all'interno delle centinaia di floppy-disks che, con inesorabile costanza, si accumulano sul tavolo.

Esistono, è vero, i **Database**, ma sono costosi ed ingombranti per il semplice compito cui sarebbero chiamati e allora conviene rimboccarsi le maniche, inserire il Compilatore preferito (**C** naturalmente) nel drive e digerirsi questa routinetta semplice semplice, ma che svolgerà più che dignitosamente il suo lavoro.

Il Programma

In sostanza ciò di cui abbiamo bisogno è una routine che legga le Directories di un disco e memorizzi i nomi dei files contenuti all'interno di un Catalogo.

Nel programma di queste pagine, il compito è svolto dal comando `Dir opt A` il cui output, però, viene rediretto su di un file temporaneo in **RAM:** così da poterlo poi esaminare con calma.

Ed ecco che entra in gioco una delle funzioni più "bestiali" dell'AmigaDOS, vale a dire quella che permette di richiamare i normali comandi **CLI** (oppure **SHELL**) dall'interno di un programma scritto in C: questa funzione si chiama **Execute()**.

```
}

/*------------------------------------------
** apre il file e lo legge carattere per carattere
** salta gli spazi bianchi e le parole indesiderate
*/
VOID
ReadFile(from_file)
FILE *from_file;
{
int i;
BYTE car, s[80], note[80];
BYTE buffer[80];
BOOL new_word=FALSE;
BOOL end_of_file=FALSE;
FILE *out_stream;

/* cerca di aprire il file Catalogo, se non lo trova ne apre uno  */
/* nuovo e lo usa per inserirvi i nomi fornita da ReadFile()      */
if(!(out_stream=fopen("Catalogo","a"))) {
    if(!(out_stream=fopen("Catalogo","w+"))) {
        PutString("Non posso aprire il file di Output\n");
        exit(0);
    }
    }
/* ripulisce il buffer una prima volta */
for(i=0;i<80;i++) {
    buffer[i]=' ';
    note[i]=' ';
    }
/* loop di lettura del file: legge un carattere per volta e lo salva
* in un buffer temporaneo
*/
while(!feof(from_file)) {
    if(new_word==TRUE)
        for(i=0;i<80;i++) {
        buffer[i]=' ';
        note[i]=' ';
    }
    /* continua a leggere finche' trova degli spazi  */
    i=0;
    do {
        car=getc(from_file); /* legge il primo carattere */
        /* e controlla che non sia la fine del file      */
        if(car==EOF) {
                            end_of_file=TRUE;
                            break;
                            }
        } while(isspace(car));
    new_word=FALSE;

    /* ha trovato un carattere usabile e lo copia nel buffer */
    do {
        /* controlla che non sia la fine del file        */
        if(car==EOF) {
                            end_of_file=TRUE;
                            break;
                            }
        sprintf(&buffer[i],"%c",car);
        i++; /* incrementa il contatore */
        car=getc(from_file); /* prende un nuovo carattere */
        } while(!isspace(car) && !end_of_file);

    if(!end_of_file) {
        /* se trova uno spazio significa che la parola e' finita    */
        buffer[i]='\0'; /* chiude il buffer con un carattere nullo */
        /* elimina tutte le ricorrenze di parole che non interessano */
        if(!strcmp(buffer,".info"))          continue;
        if(!strcmp(buffer,"Trashcan.info"))  continue;
```

...continuazione del listato in Amiga C...

Suo tramite, infatti, sarà possibile mandare in esecuzione qualsiasi programma, rintracciabile attraverso il normale path di ricerca, semplicemente dandole in pasto il nome stesso del programma o del comando CLI, passato come stringa (per un esempio vedere il listato).

A questo punto la nostra strategia dovrebbe già essere abbastanza definita:

1) tramite Execute() si richiama il comando Dir opt A rediretto sul file temporaneo RAM:[temp].

2) si apre in lettura il file ottenuto, lo si esamina e, successivamente, lo si elabora (con Sort o altri programmi)...

3) ...quindi lo si salva su disco.

Problema di... carattere

Ma qui sorge il primo problema. Infatti, come tutti avranno constatato, il risultato del comando Dir è una lista di nomi disposti su **due** colonne.

Tale visualizzazione, nella stragrande maggioranza dei casi, costituisce sicuramente un fatto positivo, ma per noi, in questo momento, risulta solamente un impiccio in più da superare, in quanto due parole in un'unica riga (anche se separate da più spazi) sarebbero lette dal programma come una sola stringa, con la conseguenza, immaginabile, di perdere una parte delle informazioni che interessano.

Il problema viene risolto dalla routine **ReadFile()** la quale si incarica di aprire il file temporaneo in lettura, suddividerne il contenuto in singole parole (per la spiegazione leggere il listato abbondantemente commentato), mostrare ogni parola così ottenuta e domandare se salvarla oppure no.

Al termine dell'operazione il comando passa al **main**, nucleo principale del programma, il quale provvede a chiamare, sempre tramite Execute(), il comando **Sort** per ordinare alfabeticamente il file **Catalogo**.

Dopodichè si avrà, su disco, un catalogo del disco esaminato da usare come meglio si crede: per la ricerca di un particolare file sarà sufficiente digitare...

```
search Catalogo qualcosa
```

```
        if(!strcmp(buffer,"Disk.info"))       continue;
        if(!strcmp(buffer,"Trashcan"))    continue;
        /* se trova la scritta (dir) la copia accanto al nome relativo */
        if(!strcmp(buffer,"(dir)")) {
                                fprintf(out_stream," %s",buffer);
                                continue;
                                }
        /* mostra il contenuto attuale del buffer */
        printf("%s\t",disk);
        printf("\x1b[33m%s\x1b[0m\n",buffer);

        if(mode==INTER) {
        /* se e' stato attivato il flag interattivo il programma */
        /* chiede se salvare il nome e se aggiungere alcune note */
        PutString("Lo salvo? (y/n/c): ");
        gets(s);
        if(*s == 'y') {
        fprintf(out_stream,"\n%s",disk);
        fprintf(out_stream,"\t%s",buffer);
        }
        else if(*s == 'c') {
        PutString("Note \x1b[33m>\x1b[0m ");
        gets(note);
        fprintf(out_stream,"\n%s",disk);
        fprintf(out_stream,"\t%s",buffer);
        fprintf(out_stream,"\t[%s]",note);
        }
        }
    /* modo di lettura automatico */
        else if(mode==DEFAULT) {
                                fprintf(out_stream,"\n%s",disk);
                                fprintf(out_stream,"\t%s",buffer);
                                }
        /* la parola e' stata copiata nel file, ne legge una nuova */
        new_word=TRUE;
        }                       /* if(!end_of_file) */
    }   /* while(!feof(from_file)) */
    fclose(out_stream); /* chiude il file di output */
}

/*-------------------------------------------
** mostra un piccolo aiuto
** Questa opzione viene attivata chiamando
** il programma seguito da punto interrogativo
*/
VOID help()
{
PutString("Uso: filcat [i][?]\n");
PutString("     i: inserimento interattivo\n");
PutString("         <y> memorizza il nome del file\n");
PutString("         <c> memorizza il nome del file con un commento\n");
PutString("         <n> salta il nome del file\n");
PutString("     ?: help\n");
PutString("-------------------------------------------------------\n");
PutString("     \x1b[33m 1991 by Alessandro Andreuccetti\x1b[0m\n");
exit(0);
}

/************ MAIN ****************************************************/
VOID
main(argc, argv)
int argc;
STRPTR *argv;
{
SHORT success;
STRPTR cmd = "Dir >RAM:[temp] ";
BYTE c[80], cont[80];
SHORT error;
/* analizza la linea di comando */
```

...continuazione del listato in Amiga C...

Precisazioni

Per funzionare, Execute() richiede la presenza del comando **Run** nella directory **C** del disco di Boot.

Se il file Catalogo non esiste nel path di ricerca impostato con la startup-sequence, il programma ne crea uno nuovo con lo stesso nome, altrimenti apre il file esistente con modalità append...

```
out_stream = fopen ("Catalogo", "a")
```

...ed aggiunge in coda i nuovi nomi, dopodiché ordina il tutto con Sort.

L'analisi della linea di comando (l'analisi, cioè, di tutto ciò che viene scritto in coda al nome del programma al momento del lancio del programma stesso) permette di capire quali siano le reali intenzioni dell'utente, per cui, impostando determinati flags, saremo in grado di stabilire se effettuare un esame interattivo del disco, oppure lasciare tutto il lavoro alla routine; se impostare il procedimento di stampa o se iniziare quello di ricerca.

Ricerca da effettuare in qualsiasi momento con un qualsivoglia programma del tipo del famosissimo **Grep**, o del già citato **Search**.

Per chi non fosse in possesso di nessun programma del genere (tra l'altro si tratta di Pubblico Dominio) niente paura: cercheremo di implementarne uno, prossimamente, proprio per il nostro Filcat.

Per chi ha fretta

Questo programma è un "semplice" archiviatore di files, utile per la ricerca veloce dei propri programmi preferiti all'interno dei molti floppys che, solitamente, compongono la dotazione standard dell'utente "medio" di computers.

L'uso di **filcat** è abbastanza intuitivo; comunque, di seguito, sono riportate le azioni da svolgere perchè funzioni correttamente.

Si può lavorare in due modi:

- **Automatico**: si accede a questa modalità semplicemente lanciando il programma e inserendo il disco da esaminare in un drive. Poi occorre specificare il nome esatto del dischetto.

```
if(argc < 2) mode = DEFAULT;/* un solo parametro: va in automatico */
else if(argc == 2)     /* due parametri: */
    switch(*argv[1]) {
        case 'i':mode=INTER; break;/* modo interattivo */
        case '?':help(); break;   /* aiuto      .      */
        default :help();
        }
do {
    PutString("Nome del disco da esaminare: ");
    gets(disk);
    strcat(cmd,disk); strcat(cmd,": opt a");
    success=Execute(cmd,0,0);
    /* apre in lettura il file ottenuto dalla redirezione di Dir */
    if(!(stream=fopen("RAM:[temp]","r"))) {
        printf("Non posso aprire il file temporaneo.\n");
        exit(0);
        }
   ReadFile(stream); /* esamina il file */
   fclose(stream); /* chiude il file di input */
   error=remove("RAM:[temp]"); /* cancella il file temporaneo */
    /* esegue il Sort del file Catalogo */
    success=Execute("Sort Catalogo Catalogo",0,0);
    PutString("\n");
    PutString("Operazione terminata con successo.\n");
    PutString("I dati salvati sono nel \x1b[33mCatalogo\x1b[0m\n");
    PutString("Vuoi salvare su disco? (y/n): ");
    gets(c);
    if(*c == 'y')
        success=Execute("Copy RAM:Catalogo DF1:",0,0);
    PutString("Vuoi esaminare un altro disco? (y/n): ");
    gets(cont);
    } while(*cont=='y');
}
```

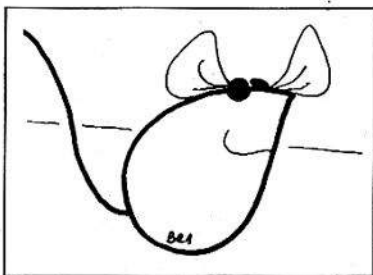
Fine del listato

- **Manuale**: per accedere a questa modalità occorre, invece, lanciare il programma seguito dalla lettera **i**, cioè

```
filcat i
```

In questo modo, dopo aver specificato il nome del disco, verrà chiesto che cosa fare riguardo il file selezionato; si potrà rispondere se salvarlo, ignorarlo o aggiungervi una nota.

- digitando il punto di domanda (**?**) si otterrà un brevissimo help.

Attenzione: per funzionare, filcat deve trovare il programma **Execute** nella directory **C** del disco di Boot.

Telefono casa...

...ma anche: come inviarci lettere, dischetti, proposte e coì via

Chi ci segue da tempo saprà certamente che mettersi in contatto con i redattori della Systems Editoriale (ed in particolare con la direzione) è la cosa più semplice di questo mondo, anche se non sempre il "contatto" risulta agevole. Per motivi organizzativi, infatti, è consigliabile telefonare (02 / 57.60.63.10) il solo giovedi pomeriggio (dalle ore 15 circa fino alle 19). L'orario fissato tiene conto, ovviamente, anche delle ridotte tariffe teleselettive vigenti in tale fascia oraria. Nessun problema, invece, con gli utenti di modem, che possono contattare la nostra BBS cui è dedicata una linea telefonica (02 / 57.60.52.11) attiva 24 ore al giorno, ma le ore notturne, a parte le considerazioni sul minimo tariffario, sono le più "pulite" e consentono trasferimenti dati senza incappare nei tipici errori dovuti a linee disturbate (viva la SIP!).

Le risposte ai quesiti posti sulla nostra BBS vengono, di solito, evasi a stretto giro di... modem; quindi cancellati. Le domande e le corrispondenti risposte che vengono ritenute di interesse generale vengono selezionate e pubblicate sulla rivista. Chi, invece, non dispone di modem e non può (o non vuole) usare il telefono, può ugualmente porre quesiti (purché "in linea" con gli argomenti trattati sulla nostra rivista) mediante lettere oppure fax (02 / 57.60.30.39) cui è dedicata una linea telefonica ed è attivo (ovviamente) anch'esso 24 ore al giorno. Le risposte, in questi due ultimi casi, non vengono mai inviate per posta (o, a maggior ragione, via fax), ma solo pubblicate sulla rivista, a patto, come al solito, che gli argomenti trattati siano ritenuti di interesse generale.

In conclusione: per avere una risposta rapida servirsi esclusivamente del telefono oppure della BBS. Chi non ha fretta si può limitare ad usare il fax oppure ad inviare, ancora più semplicemente, una lettera. In tutti i casi consigliamo, nel vostro esclusivo interesse, di specificare sempre (alla nostra centralinista, sulla busta, nella parte superiore del fax o della lettera; per la BBS è invece presente la sezione specifica) la rivista alla quale vi riferite (cioè, da settembre, Personal Computer Club). La Systems editoriale, infatti, pubblica ben dodici riviste e non è quindi sufficiente contattarci specificando semplicemente *"Spett. Systems, sono un lettore della vostra rivista... (ecc.)"*.

Per quanto riguarda la collaborazione, infine, si ricorda che in nessun caso viene restituito il materiale eventualmente inviato, nemmeno se concordato preventivamente ed, in seguito, pubblicato oppure no. A questo proposito, e qui concludiamo, è bene tener presente che le proposte di collaborazione vengono prese in esame esclusivamente se effettuate per telefono. In caso contrario si rischia di non avere la possibilità di avere risposta a proposte di collaborazione effettuate con altri mezzi (BBS, lettera, fax) come pure si rischia di spendere denaro inutilmente (per il dischetto e le spese di imballo e spedizione) per un lavoro che, difficilmente esaminabile, non risponde alle esigenze della testata.

di Filippo Bosi

Grafica in Pascal, esempi per Amiga e Ms - Dos

Tra Turbo Pascal e Kick Pascal la compatibilità può realizzarsi, anche se a costo di qualche (piccolo) sacrificio

Chi svolge abitualmente un'attività legata al mondo dei computer, quasi certamente possiede un elaboratore a casa propria, oltre che sul luogo di lavoro.

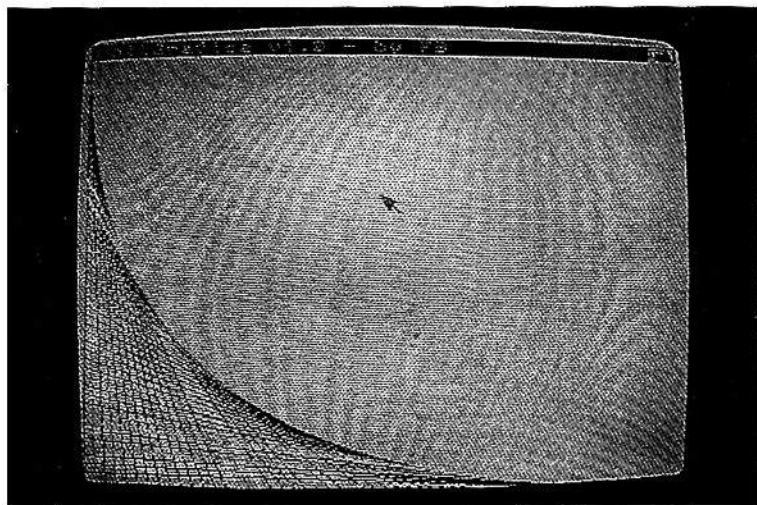
*"...Questo KickPascal (per gli amici: **KP**) assomiglia proprio tantissimo al Turbo Pascal (**TP**) che, in ufficio, è installato nel mio Ms - Dos. Chissà che non riesca a far girare quel bel programma grafico anche sul mio Amiga..."*

Forse non avrete mai osato pensare ciò, ma la somiglianza tra i due compilatori *sembra* che non possa essere sfruttata adeguatamente. Ecco, quindi, che vi verremo incontro iniziando a costruire diverse routine in grado di snellire l'arduo compito di far funzionare (sul nostro Amiga dotato di KP) programmi sorgenti scritti usando il favoloso compilatore **Borland TP 5.0**.

Grafica

Iniziamo dalla parte più piacevole ed appariscente, costruendoci una **libreria grafica** (chiamata, con molta fantasia, **TPGrafica**, vedi ultimo programma pubblicato in queste pagine) che, oltre al fatto di assomigliare notevolmente a quella TP (universalmente nota come **BGI**, **B**orland **G**raphics **I**nterface), semplifichi notevolmente la vita per l'utilizzo diretto delle librerie grafiche di Amiga, potentissime sì, ma un po' complicate per qualche semplice esperimento. Le routine in oggetto dovranno, pur mantenendo una buona compatibilità con TP, permettere di sfruttare le capacità di Amiga senza richiedere grossi sforzi da parte del programmatore. Nella stesura delle routine utilizzeremo, per quanto possibile, la convenzione di assegnare nomi identici alle procedure che eseguono gli stessi compiti delle omologhe "sotto" TP, mentre per quanto riguarda le procedure specifiche di Amiga (tra cui quelle legate alla gestione di eventi, mouse, titoli di schermi, eccetera) daremo, oltre al nome, anche un prefisso, in modo da evitare il problema di nomi duplicati in una eventuale conversione di programmi TP.

Il nome *"AttendiUnTasto"* non verrà mai utilizzato, ad esempio, per indicare procedure o funzioni del nostro insieme di routine di supporto alla conversione da TP; la probabilità che in un programma TP, da trasportare su Amiga, esista già tale identificatore di procedura non è poi così bassa. La chiameremo, magari, *"Gr_AttendiUnTasto"*, ad indicare che appartiene alla libreria di funzioni grafiche ("**Gr_**").

Pur mantenendo gli stessi identificatori nella costruzione delle nostre routine, ci troveremo spesso di fronte ad alcune difficoltà nel mantenere una **compatibilità** totale a livello di **parametri**, non tanto

legate a diversità nell'implementazione del linguaggio Pascal su Amiga, quanto proprio a diversità di *modi di operare* nei due mondi, Amiga Dos e Ms - Dos.

Esempio: come aprire uno schermo grafico che possa assomigliare a quello di un Ms - Dos? Dal momento che nel mondo Ms - Dos non esistono finestre (ad eccezione di Windows), emuleremo la grafica TP aprendo uno **schermo Intuition** con, al suo interno, una finestra priva di bordo (borderless) perché non è possibile disegnare direttamente su di uno schermo.

E' da notare come, nella definizione della procedura **InitGraph** nel sorgente di TPGrafica, nei parametri di **Open_Window** compaia la costante **Borderless**. Purtroppo dobbiamo fare subito i conti con la diversità fra Ms - Dos e Amiga. La sintassi originale di InitGraph (funzione che apre uno schermo grafico) su TP prevede la scelta fra **Driver Grafico**, **Modo Grafico** e **Path** (percorso di directories) per accedere ai Driver.

Tutto questo perché all'interno di un sistema Ms - Dos possono trovare posto svariati tipi di **schede grafiche**, ognuna caratterizzata da una propria rappresentazione in memoria delle matrici di pixel che appariranno sullo schermo e da propri comandi.

Ognuna, quindi, richiede programmi di gestione (Driver, appunto) progettati ad hoc. Ne consegue, in ambiente Ms - Dos, l'indispensabile disponibilità di un particolare Driver per ogni scheda (o quasi). Su Amiga, per fortuna, il problema non si pone. Tutti i modi grafici possibili sono supportati dal sistema operativo (più precisamente, dalle librerie graphics e intuition) e non c'è bisogno di driver differenti per ogni risoluzione di schermo da utilizzare.

Perché, quindi, complicarci la vita e simulare, per solo dovere di compatibilità con TP, una gestione complicata della grafica? Molto meno laborioso e ridondante è avere l'accortezza, peraltro minima, di adattare la chiamata ad InitGraph allo "stile" Amiga. e, di conseguenza, adattare ogni programma che importiamo da TP.

Ecco allora che, per esigenze di chiarezza, oltre a pubblicare il sorgente delle nostre routine, pubblicheremo un elenco a parte in cui specificheremo se una routine emula completamente l'omologa TP (anche a livello di tipi di parametri da passare e/o valori restituiti, nel caso di funzioni) oppure se solo a livello di comportamento; oppure, infine, se è specifica per il mondo Amiga e non ha corrispondente in TP.

La presenza di un breve programma dimostrativo delle routine, sia in versione **TP** che in versione **KP**, servirà per evidenziare ulteriormente le tecniche di conversione.

`Ctrl Q + R`	Posiziona il cursore all'inizio del testo nell'editor
`Ctrl Q + C`	Posiziona il cursore alla fine del testo nell'editor
`Ctrl + W`	Muove il contenuto dello schermo di una riga verso l'alto
`Ctrl + Y`	Muove il contenuto dello schermo di una riga verso il basso
`Ctrl Q + W`	Sposta il testo in modo da avere il cursore nella prima riga
`Ctrl Q + Y`	Sposta il testo in modo da avere il cursore nell'ultima riga
`Ctrl Q + Q`	Sposta il testo in modo da avere il cursore al centro (si ottiene anche con F8)
`Ctrl J`	Inserisce una riga vuota nella posizione del cursore
`Ctrl T`	Cancella la parola subito dopo al cursore
`Ctrl L`	Cancella un'intera riga a partire dalla posizione del cursore
`Ctrl K`	Cancella totalmente la riga dove si trova il cursore
`Ctrl Z`	Elimina una riga (cancella e sposta il testo)
`Ctrl U`	Undo - Annulla il risultato dell'ultimo comando
`Ctrl Q + F`	Ricerca di testo
`Ctrl Q + A`	Ricerca e Sostituzione
`Ctrl B + R`	Legge un blocco di testo da un file su disco
`Ctrl B + W`	Scrive un blocco di testo su di un file su disco
`Ctrl B + B`	Delimita inizio/fine di un blocco di testo (Block Mark)
`Ctrl B + S`	Sposta il cursore all'inizio di un blocco di testo selezionato
`Ctrl B + E`	Sposta il cursore alla fine di un blocco di testo selezionato
`Ctrl B + F`	Elimina la selezione di un blocco
`Ctrl B + C`	Copia il blocco selezionato nella posizione attuale del cursore
`Ctrl B + V`	Sposta il blocco selezionato alla posizione attuale del cursore
`Ctrl B + D`	Cancella il blocco di testo selezionato
`F5`	Sposta il blocco selezionato di un carattere verso sinistra
`F6`	Sposta il blocco selezionato di un carattere verso destra
`Ctrl B + P`	Stampa il blocco selezionato
`ESC`	Esce dal modo editor
`Shift + F9`	Compila il sorgente presente nell'editor
`F9`	Esegue il sorgente presente nell'editor (se necessario, compila)
`Ctrl V`	Commuta Inserimento/Sovrascrittura

I comandi dell'Editor di Kick Pascal per Amiga

Il programma dimostrativo

Costituito da pochissime righe di codice, il programma **Kpdemog1.p** apre uno schermo in bassa risoluzione (LoRes) con sei bitplanes (64 colori, quindi) e disegna una serie di linee, ognuna con colore casuale, in modo da formare un gradevole effetto geometrico.

E' interessante vedere come il programma sia indipendente dalla risoluzione dello schermo aperto. Se, infatti, si modificano i parametri della istruzione **InitGraph**, ad esempio sostituendo **Lores** (320 x 256 pixel) con **Hires** (640 x 256), oppure con **Lores + Lace** (640 x 512), il programma continua a disegnare sull'intera ampiezza dello schermo, grazie alle funzioni **GetMaxX** e **GetMaxY** che restituiscono, rispettivamente, le coordinate **X** e **Y** massime a seconda del tipo di risoluzione correntemente scelta per lo schermo grafico.

L'utilizzo delle routine permette al codice di essere utilizzato virtualmente su

qualsiasi tipo di schermo, a meno di piccolissime modifiche. Purtroppo non è sempre così semplice costruire programmi che girano indipendentemente dalla risoluzione, specialmente in caso di programmi di una certa complessità, anche se sarebbe buona regola di programmazione tener conto del fatto che il programma debba funzionare su diversi tipi di schermi, anche in vista di un (molto) probabile ampliamento della gamma di risoluzioni grafiche di Amiga.

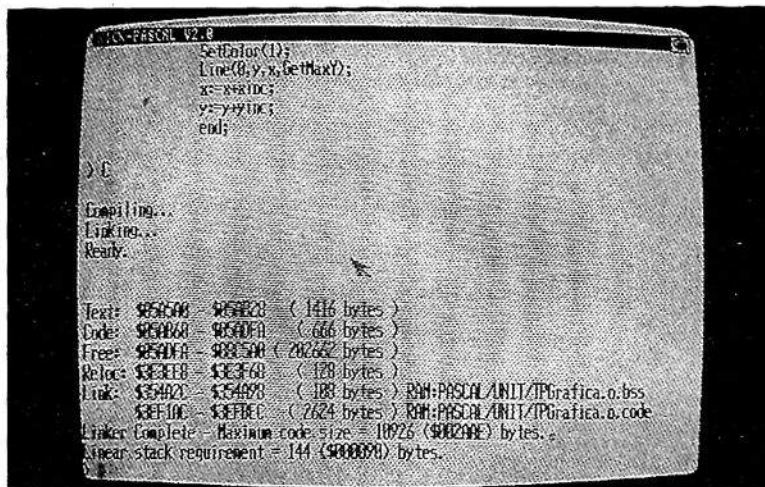
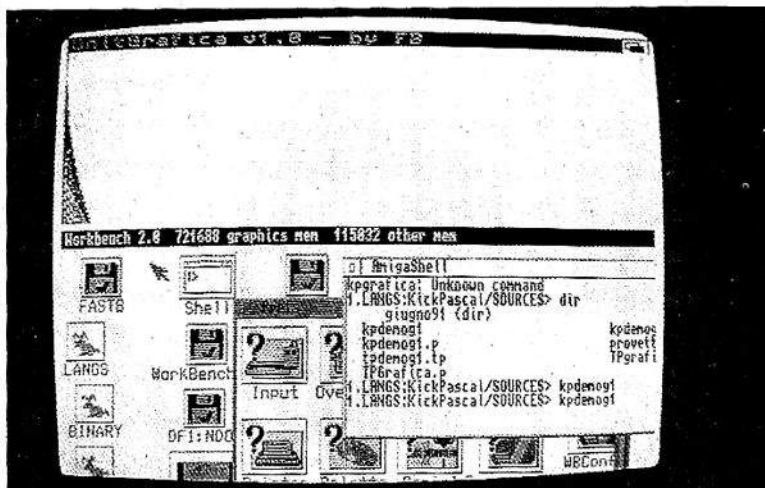
A proposito di Amiga

Amiga, almeno nei modelli attuali (non parliamo del 3000) dispone di una risoluzione massima di 640 x 512, un po' scomoda poiché interlacciata, e in grado di utilizzare contemporaneamente "solo" 64 colori (in bassa risoluzione e nel modo Extra Half Brite, cioè con 32 colori base in 2 diverse luminosità).

Bisogna quindi prestare molta attenzione per portare sotto KP un programma che apre uno schermo 320 x 200 con 256 colori, tipo di schermo comunemente utilizzato (soprattutto in ambiti.. ludici) e presente nelle schede **VGA**, ormai diffusissime nel mondo Ms - Dos.

Generalmente, a titolo informativo, le risoluzioni che presentano meno problemi per portare programmi su Amiga sono quelle tipiche della **CGA** (320 x 200 in 4 colori e 640 x 200 in 2 colori) e molte di quelle della **EGA**, pur se, in questo caso, si deve ricorrere al modo interlacciato per emulare i modi tipo 640 x 480.

Riassumendo: è inutile (o perlomeno può presentare grosse difficoltà) importare su Amiga un programma complesso che lavora ad una risoluzione di 1024 x 768 con 256 colori su IBM: sarebbe molto più semplice riprogettarlo completamente!

Teniamo, quindi, i piedi ben saldi per terra, non pretendiamo con questo insieme di routine di ottenere una compatibilità completa, garantita solo da una macchina Ms - Dos oppure dalla scheda Janus, ma solamente un modo per rendere molto più snello il passaggio di sorgenti da TP a KP, cosa sicuramente molto gradita a tantissimi studenti delle scuole superiori che possiedono un Amiga e vorrebbero provare i programmi sperimentati a scuola durante le lezioni di informatica.

1 - Digitare all'interno dell'editor KP il codice sorgente TPGrafica.p (ricordarsi di memorizzare il codice sorgente su disco... Non si sa mai !)

2 - Compilare (comando "C" se in modo linea, oppure usando i menu o Shift-F9). All'interno della directory **Unit**, se tutto è andato bene, dovrebbero trovarsi i files: "TPGrafica.u" e "TPGrafica.o".

3 - Digitare, all'interno dell'editor KP, il codice sorgente di KPDemog1.p

4 - Compilare ("C" oppure Shift-F9...)

5 - Eseguire (...finalmente !!) il programma ("R" oppure F9)

Come compilare i due programmi pubblicati

Partire con KickPascal

Per utilizzare al meglio il compilatore **Kick-Pascal** (che chiameremo, d'ora in poi, **KP**) basilare è l'ottimale configurazione tramite il programma **PascalPrefs** (d'ora in poi **PP**).

Come già detto il mese scorso, però, l'immediato utilizzo di PP è penalizzato dal fatto che sia scritto in **tedesco**(!).

Tenteremo quindi di descriverne il funzionamento in tutti i dettagli, a partire dalle scelte di configurazioni più importanti (tra cui la quantità di memoria da riservare all'editor, il tipo di schermo, i font...).

In seguito arriveremo a quelle meno basilari per un corretto funzionamento di KP sul proprio computer, ma che permettono di scegliere, fin nei minimi particolari, l'ambiente operativo più consono alle nostre esigenze (tra cui la configurazione dei tasti dell'editor).

Purtroppo la scelta di trattare le funzionalità di PP in ordine di importanza operativa costringe a saltare qua e là tra le opzioni del menu principale.

Per dovere di chiarezza, quindi, esse vengono esposte di seguito, ordinate come appaiono sullo schermo di PP ed individuate da un numero (posto a fianco) che sarà di riferimento per successive trattazioni approfondite.

Riportiamo, inoltre, la **traduzione in italiano**, non letterale, per carità, onde evitare certi obbrobri informatici, ma solo una spiegazione a livello di funzionalità.

1. Laden = carica la configurazione dal file specificato nel requester iniziale di PP.

2. Abspeichern = salva la configurazione scelta nella corrente sessione di PP nel file specificato nel requester iniziale.

3. Editor Tasten-Belegen = configurazione dei comandi dell'editor.

4. File Requester Einstellen = configurazione del file requester di KP.

5. Defaultwerte Definieren = scelta dei parametri di default dell'ambiente integrato.

6. Window Einstellungen = configurazione del modo video (grandezza finestra, tipo di schermo) del compilatore.

7. ProgrammEnde = uscita dal programma PP.

Premessa importantissima: ogni scelta da menu principale (eccetto le 1, 2, 7) porta ad una schermata con 3 "bottoni" standard (gadget) posizionati nella parte bassa dello schermo che sono, da sinistra verso destra, Fertig, Hilfe, Rucksetzen; ecco il loro significato:

1. Fertig = (ok) accetta i dati contenuti nello schermo attualmente visualizzato come validi e ritorna al menu principale di pp.

2. Hilfe = (Help) visualizza uno schermo di aiuto (in tedesco...) per meglio(..) interpretare i dati richiesti.

3. Rucksetzen = (Forget It) ritorna al menu principale annullando eventuali modifiche apportate ai parametri presenti attualmente sullo schermo.

Detto ciò, passiamo a descrivere le singole voci del menu principale, assumendo che al fine di un corretto utilizzo delle voci 1, 2, 7 sia sufficiente la descrizione data sopra, con l'unica raccomandazione di **non** uscire dal programma PP (scelta 7) senza prima aver **salvato** la configurazione (voce 2), altrimenti avremo lavorato per niente, e il programma KP continuerà a "comportarsi", come se niente fosse, nè più nè meno che come nella sessione di lavoro precedente l'utilizzo di PP.

6. Window Einstellungen: configurazione delle opzioni di visualizzazione.

Nell'ordine, dall'alto verso il basso, appaiono, dopo la scelta 6 da menu, le seguenti voci:

- Screen:

Puntare il mouse e premere il bottone sull'opzione scelta:

Wb = l'editor KP si presenterà come finestra sullo schermo del WorkBench; è una opzione utile per chi dispone di poca memoria libera, penalizzando però la velocità operativa (si ha uno scrolling abbastanza lento del testo).

Custom = l'editor KP ha un proprio schermo custom, indipendente da quello del WorkBench: massima velocità di gestione del testo, specie se sul WorkBench sono aperte altre finestre.

Può dare qualche fastidio nel debugging dei programmi, poiché generalmente le finestre di questi ultimi vengono aperte sullo schermo del WorkBench.

- Fenstername:

Di fianco si può scrivere il nome della finestra (nel caso si sia scelto *WB* nell'opzione precedente) o dello schermo (nel caso si sia scelto *Screen*) dell'ambiente integrato KP (Opzione simpatica dal punto di vista estetico, ma di dubbio valore pratico).

- Position X:

Coordinata X, cioè orizzontale, che rappresenta la posizione della finestra dell'editor all'interno dello schermo all'apertura del programma KP (ovviamente può successivamente essere spostata con il mouse). Nel caso si scegliesse *Custom Screen* nella prima opzione, è bene rispondere **0**, per sfruttare tutto lo spazio a disposizione.

- Position Y:

Posizione della finestra dell'editor (coordinata verticale), con le stesse considerazioni della voce precedente.

- Breite:

Larghezza, in pixel, della finestra; si consiglia di impostare 640 se si utilizza uno schermo custom.

- Hohe:

Altezza, in pixel, della finestra; impostare 256 se si utilizza uno schermo custom.

- Zeichensatz:

Font da utilizzare in ambiente KP. Bisogna specificarne il nome *senza* il suffisso ".font"; se si volesse, ad esempio, utilizzare il font **siesta.font**, si dovrebbe scrivere semplicemente "siesta". Ovviamente il font specificato deve trovarsi nella directory *fonts:* di sistema e **non** deve essere un font proporzionale (vanno quindi bene *siesta, pearl, pcfont*). Nel caso il font non fosse trovato o non fosse adatto per la visualizzazione su editor KP (ad es. font proporzionale, tanto per insistere), KP sceglie automaticamente il font di default di

sistema (generalmente *topaz.font*), quindi via libera, senza problemi, alla ricerca del font migliore per le proprie esigenze...

- **Zeichenhohe**:

Grandezza, in punti, del font desiderato. Generalmente per i font non proporzionali si hanno a disposizione 8, 9 e 11 punti: 8 = visualizzazione ad ottanta colonne; 9 = visualizzazione a sessanta colonne; 11 = caratteri molto grandi, riposano la vista, ma fanno impazzire se dobbiamo lavorare su programmi molto lunghi, in quanto solo poche righe di programma sono visualizzabili contemporaneamente sullo schermo. Le considerazioni appena fatte sono valide per chi lavora su uno schermo 640 x 200. Se non viene trovata la grandezza richiesta, KP assume il font di default.

Se, quindi, scegliamo ad esempio siesta 11, mentre nella nostra directory fonts: è presente solo siesta 8, KP visualizzerà i testi in topaz 8 (o nel font di default di sistema se diverso, per i possessori di WB 2.0).

```
*****************************************
* Sintassi e descrizione delle procedure *
* e funzioni rese disponibili dalla Unit *
* TPGRAFICA (v1.0)                       *
*                                        *
* per      . COMPUTER CLUB               *
*          by Filippo Bosi               *
*****************************************

-----------------------------------------
(TURBO)(PARAMETRI)

Procedure ClearDevice;

Pulisce lo schermo grafico
-----------------------------------------
(TURBO)(PARAMETRI)

Procedure CloseGraph;

Chiude lo schermo grafico
-----------------------------------------
(TURBO)(PARAMETRI)

Function  GetColor:word;

Restituisce il Colore di penna Corrente.
-----------------------------------------
(TURBO)(PARAMETRI)

Function  GetMaxX:integer;

Restituisce il valore max di coordinata x
reso disponibile dallo schermo grafico
attualmente aperto.
-----------------------------------------
(TURBO)(PARAMETRI)

Function  GetMaxY:integer;

Restituisce il valore max di coordinata y
reso disponibile dallo schermo grafico
attualmente aperto.
-----------------------------------------
(AMIGA)

Procedure Gr_AttendiUnTasto;

Attende fino a quando non viene premuto un
tasto (oppure un bottone di mouse) nella
finestra grafica.
-----------------------------------------
(AMIGA)

Procedure Gr_CloseLibs;

Chiude le librerie di sistema necessarie
per il funzionamento della Unit. DEVE
ESSERE ESEGUITA ALLA FINE DEL PROGRAMMA
CHE UTILIZZA LA UNIT.
-----------------------------------------
(AMIGA)

Procedure Gr_OpenLibs;

Apre le librerie necessarie al corretto
funzionamento della Unit. DEVE ESSERE
UTILIZZATA ALL'INIZIO DEL PROGRAMMA CHE
RICHIAMA QUESTA UNIT.
-----------------------------------------
(AMIGA)

Procedure Gr_SetGraphTitle(Titolo:string);

Dà un nuovo titolo alla finestra grafica.
-----------------------------------------
(TURBO)(PARAMETRI)

Function  GraphResult:integer;

Restituisce event. codice d'errore della
ultima istruzione grafica eseguita.

VALORI DI RITORNO:
GrOk    = TUTTO OK.
GrError = ERRORE GENERICO.
-----------------------------------------
(TURBO)

Procedure InitGraph(modo,npiani:integer);

Inizializza lo schermo grafico.

PARAMETRI:

modo = modo grafico
 è una combinazione delle seguenti
 costanti:
```

```
GR_LORES = bassa risoluzione (320 pixel)
GR_HIRES = alta risoluzione  (640 pixel)
GR_LACE  = interlacciato

npiani = numero piani di bit
 indica il numero di piani di bit da
 riservare allo schermo grafico.

 es. 2 piani di bit = 4 colori
 ci sono limitazioni al numero di piani:
 GR_LORES -- max 6 piani di bit (64 col)
 GR_HIRES -- max 4 piani di bit (16 col)
-----------------------------------------
(TURBO)(PARAMETRI)
Procedure Line(x1,y1,x2,y2:integer);
Traccia una linea dal punto di coordinate
(x1,y1) al punto (x2,y2). Usa il colore di
penna corrente.
-----------------------------------------
(TURBO)(PARAMETRI)
Procedure SetColor(Colore:word);
Setta un nuovo colore di penna.
-----------------------------------------
N.B. in testa alla sintassi di ogni
     funzione appaiono i seguenti
     identificativi:
-----------------------------------------
     (TURBO) -- la procedura/funzione
                svolge un compito analogo
                alla omologa funzione
                presente in TURBO PASCAL
   + (PARAMETRI) -- compatibilità
                    anche a livello di
                    parametri con TP.
   + (RISPOSTA)  -- compatibilità a
                    livello di ritorno
                    valore (nel caso di
                    funzioni).
-----------------------------------------
     (AMIGA) -- funzione specifica della
                implementazione su Amiga.
-----------------------------------------
```

```
program Vale;
{ -------------------------------- }
{ KPDEMOG1.p                       }
{ dimostrativo per l'utilizzo della }
{ unit TPGRAFICA.  By Filippo Bosi }
{            (per Amiga)           }
{ -------------------------------- }
{ VERSIONE KICKPASCAL              }
{ -------------------------------- }

  uses TPGrafica;

  var
    ErrCode  : Integer;
    xinc,yinc,
    i,x,y    : Integer;

  const
    numlinee = 50;

  begin
    { * apri le librerie grafiche * }
    Gr_OpenLibs;


    { * schermo a bassa risoluzione * }
    { * con 6 piani di bit, cioe'   * }
    { * 2^6=64 colori --> Extra     * }
    { *                   HalfBrite * }
    InitGraph(GR_LORES,6);
    ErrCode := GraphResult;

    { * se e' tutto Ok, allora disegna * }
    if ErrCode = grOk then
      begin
      xinc:=round(GetMaxX/numlinee);
      yinc:=round(GetMaxY/numlinee);
      x := 0;
      y := 0;

      for i:=0 to numlinee do
               begin
               SetColor(Random(64));
               Line(0,y,x,GetMaxY);
               x:=x+xinc;
               y:=y+yinc;
               end;

       { * attendi che venga premuto un * }
       { * tasto oppure un bottone del  * }
       { * mouse sullo schermo grafico  * }
       Gr_AttendiUnTasto;

       { * chiudi lo schermo grafico * }
       CloseGraph;

      end
    else
      WriteLn('Errore in InitGraph');

    { * chiudi le librerie * }
    Gr_CloseLibs;
end.
```

Il programma dimostrativo

```
program Gino;
{ ---------------------------------- }
{ TPDEMOG1.tp                        }
{ dimostrativo grafico               }
{ - notare le differenze con la      }
{   versione KickPascal              }
{ ---------------------------------- }
{ VERSIONE TURBO PASCAL V5.0         }
{ ---------------------------------- }

  uses Graph;

  var
    grDriver : Integer;
    grMode   : Integer;
    ErrCode  : Integer;
    xinc,yinc,
    i,x,y    : Integer;

  const
    numlinee = 50;

  begin
    grDriver := Detect;
    InitGraph(grDriver,grMode,'');
    ErrCode := GraphResult;
    if ErrCode = grOk then
      begin
  xinc:=round(GetMaxX/numlinee);
  yinc:=round(GetMaxY/numlinee);
  x := 0;
  y := 0;

  for i:=0 to numlinee do
                      begin
                      SetColor(Random(16));
                      Line(0,y,x,GetMaxY);
                      x:=x+xinc;
                      y:=y+yinc;
                      end;
      ReadLn;
      CloseGraph;
    end
   else
     WriteLn('Errore in InitGraph');
 end.
```

La versione Turbo Pascal Borland del programma dimostrativo

La Unit di supporto per gestire la grafica in Kick Pascal

```
Unit TPGrafica;

{ ------------------------------------ }
{  TPGRAFICA.p                         }
{                                      }
{  Unit di supporto per la grafica tipo }
{  Turbo Pascal v.4.0/5.0 Ms-Dos       }
{                                      }
{             by     Filippo Bosi      }
{ ------------------------------------ }


Interface

const
 GrOk = 0;  { Tutto OK }
 GrError = -1;  { Errore    }

{ costanti per InitGraph: modo grafico }
 GR_LORES = 1;
 GR_HIRES = 2;
 GR_LACE  = 4;

{ **************************** }
{ * procedure simili a quelle * }
{ * del TURBO PASCAL         * }
{ **************************** }

{ * stessa sintassi e funzionamento * }

Procedure CloseGraph;
Procedure SetColor(colore:word);
Procedure Line(x1,y1,x2,y2:integer);
Procedure ClearDevice;
Function  GraphResult:integer;
Function  GetMaxX:integer;
Function  GetMaxY:integer;
Function  GetColor:word;

{ * stesso funzionamento ma sintassi * }
```

```
{ * differente                        * }

Procedure InitGraph(modo,npiani:integer);

{ * procedure specifiche per AMIGADOS * }
{ * (precedute da Gr_ per evitare     * }
{ *  nomi duplicati con altri         * }
{ *  programmi, magari portati da     * }
{ *  Turbo Pascal.....)               * }

Procedure Gr_SetGraphTitle(titolo:string);
Procedure Gr_OpenLibs;
Procedure Gr_CloseLibs;
Procedure Gr_AttendiUnTasto;

Implementation
{$incl "graphics.lib"}
{$incl "intuition.lib"}

var
{ * titolo schermo grafico * }
TitoloSchermo:string;
{ * dimensioni finestra grafica in pixel * }
MaxX,MaxY: integer;
{ * colore penna corrente * }
ColoreCorrente: word;
{ * risultato dell'ultima operazione grafica * }
GrResult: integer;


 rp:p_RastPort;
 Win:p_Window;
 Scr:p_Screen;

Procedure InitGraph;
{ * apre lo schermo grafico * }

var
 risx,risy : integer;
 maxplanes : integer;
 scrmode   : word;
 { * dimensioni finestra * }
 fin_largh,fin_alt : integer;

const
 { * altezza in pixel del titolo * }
 { * dello schermo grafico       * }
 SCREENBAR = 10;

begin
scrmode:=GENLOCK_VIDEO;

if (modo and GR_LORES)<>0 then
 begin
 maxplanes:=6;
 risx:=320;

  { * se si richiedono 6 bitplanes * }
  { * in bassa risoluzione, allora * }
  { * vai in modo 64 colori (EHB)  * }
 if npiani=6 then
  scrmode:=scrmode+EXTRA_HALFBRITE;
 end
   else begin
 maxplanes:=4;
 risx:=640;
 scrmode:=scrmode+HIRES;
        end;

{ * controlla se si è scelto il modo * }
{ * interlacciato                    * }
if (modo and GR_LACE)<>0
       then begin
             risy:=512;
             scrmode:=scrmode+LACE;
             end
       else risy:=256;

{ * controlla che il numero dei piani * }
{ * di bit non sia superiore al max.  * }
{ * consentito                        * }
if npiani>maxplanes then
 begin
  GrResult:=GrError;
 exit;
 end;

Scr:=Open_Screen(0,0,risx,risy,
                 npiani,3,1,
                 scrmode,
                 ^TitoloSchermo);

{ * controlla che lo schermo sia * }
{ * stato aperto correttamente   * }
if Scr=NIL then
 begin
 GrResult:=GrError;
 exit;
 end;

fin_largh:=risx;
fin_alt:=risy-SCREENBAR;

Win:=Open_Window(0,SCREENBAR,fin_largh,
                 fin_alt,1,0,BORDERLESS+
                 GIMMEZEROZERO+
                        ACTIVEWINDOW,
                 Nil,Scr,fin_largh,
                 fin_alt,fin_largh,
                 fin_alt);

{ * controlla che la finestra sia * }
{ * stata aperta correttamente    * }
if Win=NIL then
 begin
 GrResult:=GrError;
 exit;
 end;

rp:=Win^.Rport;

{ * sistema i contenuti delle          * }
{ * variabili globali per dimensioni  * }
{ * schermo grafico                    * }
MaxX:=fin_largh-1;
MaxY:=fin_alt-1;
```

```
{ * sistema il colore della penna * }
{ * corrente                      * }

SetColor(1);
end;

Procedure ClearDevice;
{ * cancella il contenuto della * }
{ * finestra grafica            * }

begin
 ClearScreen(rp);
end;

function GetMaxX;
{ * ritorna il massimo valore * }
{ * della coordinata x per il * }
{ * tipo di schermo grafico   * }
{ * correntemente aperto      * }


begin
 GetMaxX:=MaxX;
end;

function GetMaxY;
{ * ritorna il massimo valore * }
{ * della coordinata y per il * }
{ * tipo di schermo grafico   * }
{ * correntemente aperto      * }
begin
 GetMaxY:=MaxY;
end;

procedure Line;
{ * traccia una linea * }
begin
 Move(rp, x1,y1);
 Draw(rp, x2,y2);
end;

procedure SetColor;
{ * setta il colore della penna * }
begin
 ColoreCorrente:=colore;
 SetAPen(rp,colore);
end;

function GetColor;
{ * ritorna il colore corrente * }
{ * della penna                * }

begin
 GetColor:=ColoreCorrente;
end;

function GraphResult;
{ * ritorna un eventuale codice * }
{ * di errore dell'ultima       * }
{ * routine grafica eseguita    * }

begin
 GraphResult:=GrResult;
end;

procedure CloseGraph;
{ * chiude lo schermo grafico * }

begin
 Close_Window(Win);
 Close_Screen(Scr);
end;

procedure Gr_SetGraphTitle;
{ * da' un nuovo titolo allo * }
{ * schermo grafico          * }

begin
 TitoloSchermo:=Titolo;
 if (Scr<>NIL) and (Win<>NIL) then
   SetWindowTitles(Win,Ptr(-1),
             ^TitoloSchermo);
end;

procedure Gr_OpenLibs;
{ * apre le librerie "graphics" e * }
{ * "intuition" (da eseguire allo * }
{ * inizio di ogni programma che  * }
{ * utilizza UnitGrafica)         * }

begin
OpenLib(IntBase,"intuition.library",0);
OpenLib(GfxBase,"graphics.library",0);
end;

procedure Gr_CloseLibs;
{ * chiude le librerie "graphics" e * }
{ * "intuition" - da eseguire alla  * }
{ * fine di ogni programma che      * }
{ * utilizza UnitGrafica            * }

begin
CloseLib(IntBase);
CloseLib(GfxBase);
end;

procedure Gr_AttendiUnTasto;
{ -- ritorna quando è stato premuto  -- }
{ -- un tasto nella finestra grafica -- }
{ -- (oppure un tasto del mouse...)  -- }
var Msg:^IntuiMessage;


begin
 ModifyIDCMP(Win,RAWKEY+MOUSEBUTTONS);
 Msg:=Wait_Port(Win^.UserPort);
 Msg:=Get_Msg(Win^.UserPort);
 Reply_Msg(Msg);
 ModifyIDCMP(Win,0);
end;

begin
Scr:=NIL;
Win:=NIL;
TitoloSchermo:='UnitGrafica v1.0 - by FB';
end;
```

di Luigi Callegari

I migliori Assembler 68000 per Amiga

Diamo uno sguardo ai vari Assemblatori disponibili per il nostro Amiga

Iniziamo, da questo mese, una serie di articoli riguardanti la programmazione in linguaggio **Assembler di Amiga**, che seguirà la falsariga della serie già presenta da alcuni mesi dal nostro Giancarlo Mariani per i processori della famiglia Intel 80X86, cuori pulsanti dei PC-compatibili.

Ricordiamo che la linea di computer Commodore Amiga, (A-500, A-1000, A-2000 ed A-3000) monta processori della famiglia Motorola 68000 utilizzati, tra l'altro, in molte workstation professionali.

Il linguaggio Assembler

Programmare in linguaggio macchina è talvolta l'unico modo per risolvere certi problemi pratici. L'Assembler, ricordiamo, è il linguaggio di più basso livello utilizzabile da un programmatore: più in basso vi sarebbe soltanto lo specificare direttamente le sequenze di zeri ed uni binari usati dal microprocessore!

Se il **linguaggio C** rimane il principe nel mondo Amiga per lo sviluppo di software (basta considerare, con una stima realistica, che circa il 90% del software commerciale per Amiga è stato scritto con questo linguaggio) l'Assembler rimane tuttavia l'unica risorsa in vari campi di applicazione.

Ad esempio, quando si deve lavorare molto "vicino" all'hardware della macchina, controllare cioè direttamente il **Copper** od il generatore sonoro di Amiga, oppure quando le applicazioni sono da calcolare con precisione nei tempi di esecuzione di ogni istruzione per motivi di sincronismo o di ottimizzazione nella velocità, non resta che ricorrere all'Assembler.

Del resto nulla vieta di usare linguaggi di livello più alto per scrivere le parti meno critiche e più ripetitive di un programma, come ad esempio l'interfaccia grafica o la gestione delle opzioni specificate sulla linea dello Shell, per poi realizzare routine in assembly da affiancare a questi moduli scritti a livello più alto, dove svolgere con la massima efficienza possibile le operazioni più critiche del software.

In ambiente Amiga è estremamente facile interfacciare parti in linguaggio macchina con porzioni scritte in altri linguaggi (**GFA Basic**, linguaggio **C**, **Modula-2**...).

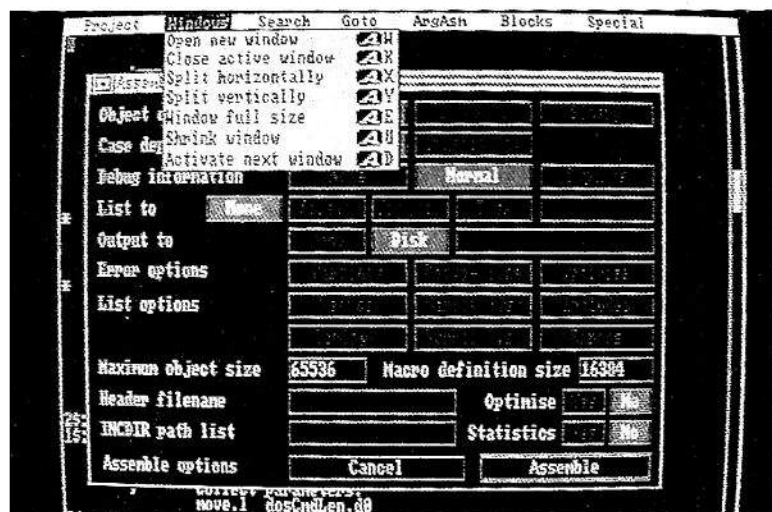
I pacchetti

Un Assembler non è, basilarmente, un programma complesso da realizzare nello stadio più semplice.

In effetti, esistono, per Amiga, alcuni assemblatori di pubblico dominio, funzionanti da Shell/CLI, che svolgono egregiamente il proprio lavoro, come ad esempio **A68K** (di **prossima pubblicazione** su AmiGazzetta).

Tra i prodotti commerciali, solitamente più completi e con vari programmi di supporto, troviamo parecchi nomi: **DevPac**, **Cape**, **AssemPro**, **Adapt**, **Metacomco**, **ArgAsm**, **Seka** ed altri ancora, come ad esempio quelli inseriti nei pacchetti dei compilatori **Aztec C** e **Lattice SAS/C**.

Metacomco

Metacomco Assembler è stato storicamente il primo macroassembler disponibile per Amiga.

I processori della famiglia Motorola 68000

Vediamo brevemente le caratteristiche dei processori della famiglia **Motorola 68000**.

Da notare che, a parità di clock e di caratteristiche di base interne, i processori della famiglia **Motorola** sono notoriamente più evoluti e veloci di quelli **Intel**, i quali sono costruiti *dovendo* rispettare la compatibilità con i vecchi **8088** e **8086**, dunque con parallelismo hardware (registri interni) e software (set di istruzioni) quasi inesistente e gestione della memoria farraginosa e lenta (tutti sanno i problemi di gestire più di 640K sotto Ms - Dos, mentre un Amiga 500 con processore 68000 non ha problemi a gestire anche 8 megabyte di memoria in linea).

Viene da chiedersi sino a quando la Intel sarà costretta a costruire microprocessori potenzialmente sempre meno efficienti di quelli concorrenti prodotti da altre case (Motorola) per la necessità di mantenere questa compatibilità con processori concepiti dieci anni fa (quando 64K di memoria erano un patrimonio, ad esempio), sprecando così quella velocità e potenza che il progresso tecnologico potrebbero consentire.

❑❒

MC68000

Prodotto nel **1979** ed inserito negli Amiga 500, 1000 e 2000, dispone di un **bus indirizzi** a **24** bit (indirizza direttamente 16 megabyte di RAM senza segmentazioni, come avviene per l'Intel 8086, che vede la memoria a blocchi di 64K!) e **di 14 registri** di uso generale perfettamente simmetrici.

La ALU è ampia **16 bit** e le parole di istruzione possono essere lunghe da **1** a **5** word di 16 bit.

Il 68000 è strutturalmente molto più flessibile e potente dell'8086, essendo "nato" internamente a 16 bit senza problemi di compatibilità verso il basso con processori ad otto bit (Intel 8088) e la sua struttura ad alta simmetria, nonchè alcune caratteristiche di funzionamento hardware (come il single-step "integrato") lo hanno reso uno dei processori più usati per vario tempi in schede a microprocessore dedicate.

68008

Versione ridotta del 68000, con capacità di indirizzamento a 20 o 22 bit (1 o 4 Mb), montato sul vecchio **Sinclair QL**, ideale per montaggi dove sono necessari semplificazioni circuitali. E per il resto identico al 68000.

68010

Rispetto al capostipite 68000, introduce il "loop mode", ovvero è in grado di eseguire più velocemente brevi cicli di istruzioni, grazie ad un piccolo "cache" che riduce gli accessi alla memoria esterna.

Inoltre, è fornito di una rudimentale MMU (memory management unit), che gli consente, ad esempio, di proseguire l'esecuzione di un programma dopo avere incontrato una istruzione errata per difetto della memoria.

Infine prevede una istruzione supplementare ed alcune differenze nelle classi delle istruzioni (supervisore ed utente).

68012

In pratica, a parte differenze di contenitore, è il 68010 con il bus esteso per gestire **16Mb** o **2 Gb** di memoria.

68020

Bus dati a **32 bit** (capacità di indirizzamento pari a **4 Gb** in linea) e di lettura di una intera longword (32 bit) dalla memoria con **un solo ciclo di clock** sono il biglietto di presentazione del primo "full 32 bit" della famiglia.

La CPU prevede anche 256 byte di memoria interna rapidissima (cache) che gli consente di ottimizzare l'esecuzione di cicli di istruzioni, nuovi modi di indirizzamento della memoria (supplementari a quelli del 68000 e "sovrapposti", senza problemi di compatibilità verso il basso) e di manipolazione dei bit, matematica BCD, supporto hardware per sistemi multiprocessore e per coprocessori matematici **68881/2**, visti come periferiche che aggiungono istruzioni alla CPU.

68030

Montato di serie su molte schede acceleratrici per Amiga 2000 e sul recente **Amiga 3000**, è un 68020 con PMMU (page memory management unit, equivalente al chip esterno **68851**), un maggiore numero di registri interni speciali, due cache (istruzioni e dati) da 256 byte, maggiore velocità di esecuzione delle istruzioni.

68040

Già protagonista delle più recenti schede acceleratrici per Amiga 3000 e Amiga 2000, probabile cuore del **futuro Amiga 4000**, è il microprocessore più potente e sofisticato in tecnologia CISC (Complex Instructions Set Computer) attualmente sul mercato.

Composto da 1.200.000 transistor, lavora con clock da 25 Mhz sino a **50 Mhz** (in future versioni), esegue i benchmark interi a **20** MIPS (contro **i** 15 dell'Intel 80486) e quelli in virgola mobile a 3,5 MFlops (contro lo 1,0 dell'Intel 80486), termina mediamente le istruzioni in 1,3 cicli di clock.

Dispone inoltre di una FPU incorporata di prestazioni superiori al 68882, ha due cache da 4096 byte ciascuno per istruzioni e dati e due MMU in grado di trasmettere i dati in parallelo (non serialmente come avviene per il 68030 e l'80486) raggiungendo un vertiginoso 200 Mbyte al secondo di velocità di traferimento di picco.

Fu infatti fornito insieme ai primissimi modelli di **Amiga 1000**, negli USA, agli sviluppatori delle varie softhouse che si erano impegnati con la Commodore a costruire una base software per la nuova macchina.

Per questo motivo ad esso si fa spesso riferimento come assemblatore standard, anche perchè si adeguava perfettamente ai dettami della Motorola, prevedendo cioè tutte le parole riservate e le direttive nel formato preciso con il quale erano state definite dalla casa costruttice del processore 68000.

In effetti, il Metacomco Assembler era un semplice **assembler di linea**, estremamente lento ed incapace di linkare direttamente le funzioni di libreria invocate nel modulo oggetto, compito per il quale si doveva usare, sempre da Shell, il linker **ALink**, dalla lentezza esasperante.

Seka

Seka Assembler è un vecchio assemblatore prodotto dalla **Kuma**, un softhouse ben nota per avere prodotto software di qualità per vecchie macchine ad otto bit come il **Sinclair Spectrum** ed il **Commodore 64**.

E stato probabilmente il primo assemblatore a rendere disponibile un rudimentale ma pratico sistema "integrato", dove cioè il programmatore poteva redigere il testo da compilare, avviare la compilazione e chiamare un **monitor disassembler** restando praticamente sempre all'interno di un solo programma, che funge da interfaccia. Con il Seka sono stati prodotti moltissimi programmi di pubblico dominio, in particolare da hacker e "smanettoni", essendo di basso costo, semplice da usare, con scarse esigenze di memoria e, appunto, abbastanza bene integrato (e sino a qualche tempo fa era l'unico ad esserlo).

Il rovescio della medaglia è che il Seka non rispetta molto rigidamente le direttive Metacomco, introducendo direttive personalizzate e varianti di quelle standard che rendevano il programma "non trasportabile" immediatamente sotto altri assemblatori standard. Inoltre, l'ambiente integrato è piuttosto rudimentale, funzionando esclusivamente da tastiera e non usando nessuna delle facilitazioni di interfacciamento consentite da Intuition.

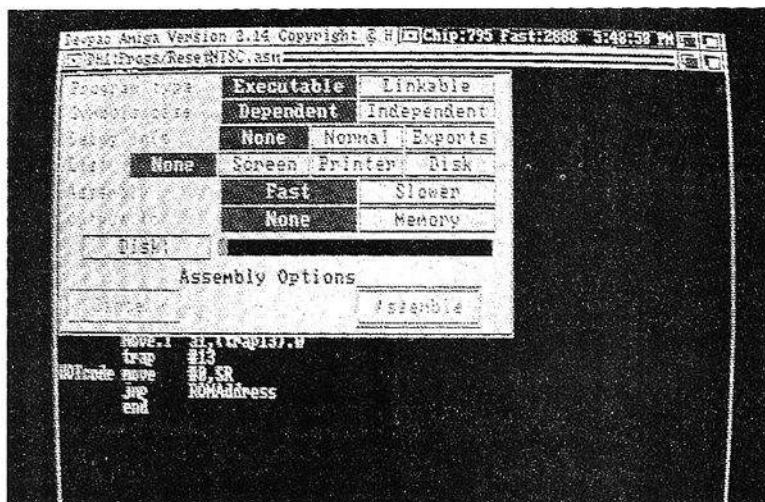
Oggigiorno il Seka viene comunque usato, per programmi abbastanza brevi, da molti aficionados che non vogliono passare a programmi più completi ed evoluti od occupanti troppa memoria.

AssemPro

Sviluppato dalla tedesca **Data Becker**, AssemPro è un bel programma integrato, dotato di editor ed assembler (con estensione per il processore **68010**) e di alcune utility parallele molto interessanti.

Il programma, piuttosto lungo, usa completamente **Intuition** e fornisce menu, gadget, supporto al **mouse** sia nell'editor, sia nell'efficiente monitor debugger (che consente, ad esempio, di delimitare la parte di codice da eseguire evidenziandola col mouse) sia nei programmi di supporto.

Rilasciato in lingua inglese e tedesca, AssemPro è stato usato a lungo in passato e viene tuttora sfruttato diffusamente in Germania: chi acquista riviste teutoniche trova regolarmente routine assembler studiate per AssemPro, che laggiù è un po' lo standard degli assemblatori.

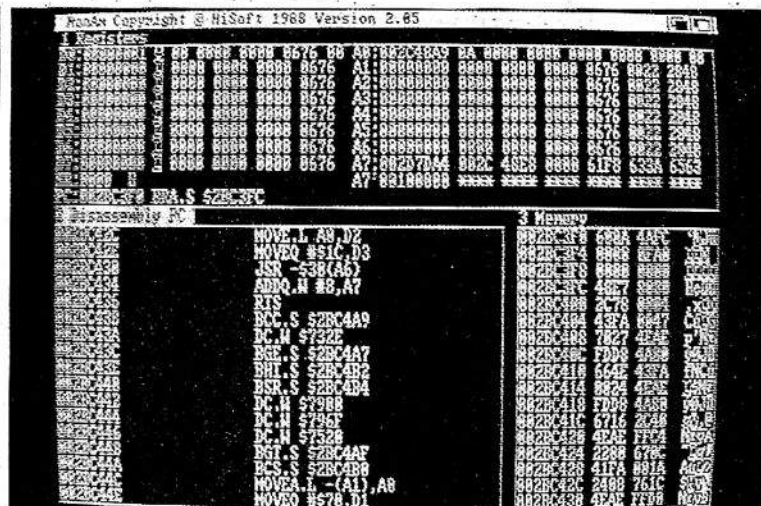
In effetti, AssemPro non collima perfettamente con le direttive Metacomco e,

quindi, i listati scritti appositamente per esso sono quasi sempre da rimaneggiare prima di essere compilati con altri pacchetti. Inoltre, AssemPro non è in grado di produrre moduli oggetto con linking separato (non è neppure fornito un linker indipendente, come per tutti gli altri pacchetti moderni), per cui è inutile per realizzare moduli assembly da inserire agevolmente e comodamente (in fase di linking) con moduli oggetto scritti con altri linguaggi.

Inoltre, la documentazione con il quale veniva fornito era piuttosto rozza e imprecisa, in particolare nell'edizione inglese (dalla Abacus) del pacchetto.

DevPac

Prodotto dalla inglese **HiSoft**, è attualmente il pacchetto preferito da molti utenti.

Tra l'altro, è l'unico che fornisce un sistema integrato Editor/Assembler/Debugger molto versatile e veramente completo e che è uscito in **edizione completamente italiana** (manuale e programma), pubblicato dalla **Leader** di Casciago.

DevPac aderisce perfettamente allo standard Motorola, può generare moduli eseguibili o linkabili in un secondo tempo, può svolgere un insieme di ottimizzazioni automatiche sul codice generato, introvabili in qualunque altro assemblatore noto ed implementa varie estensioni alle direttive standard Motorola.

L'editor fornito, dotato di gadget, requester e menu in stile Intuition, può essere utilizzato per redigere, con varie opzioni di editing tipiche degli editor di linea, il testo sorgente, ma anche per lanciare l'assemblatore (che è comunque richiamabile anche da Shell come comando separato) od il monitor.

Il monitor debugger è dotato, oltre che di tutte le funzioni tipiche di un programma di questo genere (breakpoint, esecuzione passo passo e per un certo numero di cicli, visualizzazione su più finestre di aree di memoria e contenuti dei registri, disassemblaggio intelligente...) anche di una limitata capacità di decodifica delle istruzioni di Amiga che fanno riferimento a funzioni di libreria, sostituendo agli offset di chiamata direttamente il nome della funzione invocata.

DevPac, giunto alla versione **2.14**, è il pacchetto che useremo abitualmente nelle nostre chiacchierate sull'Assembler.

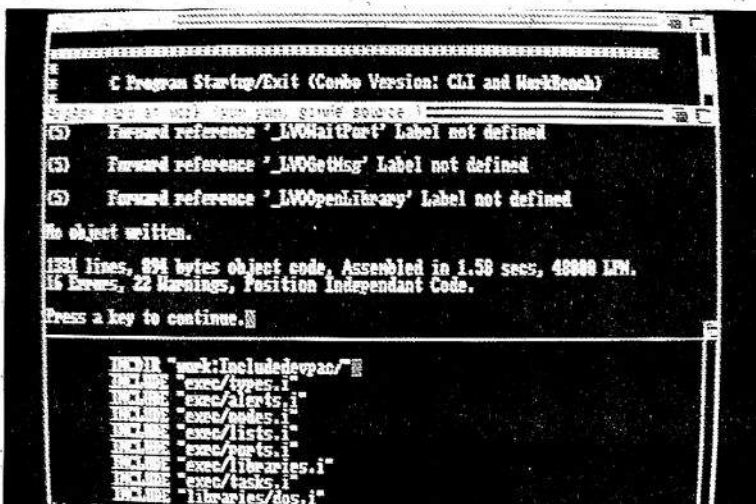
ArgAsm

Prodotto dalla inglese **Argonaut** e scritto da Jez San, autore di alcuni videogiochi commerciali per Ms - Dos ed Amiga, questo assemblatore è stato scritto per avere la massima velocità possibile di assemblaggio.

Funziona in singola passata per la maggior velocità possibile, ha un discreto rispetto degli standard Metacomco, dispone di interfaccia intuitionizzata con supporto per mouse menu e gadget e l'ambiente Assembler/Editor è notevolmente integrato.

Viene fornito senza linker, ma può produrre programmi linkabili a parte e codici binari puri o direttamente codici eseguibili da AmigaDOS ed è in grado di svolgere alcune ottimizzazioni sul codice. Non è fornito con alcun debugger e disassemblatore.

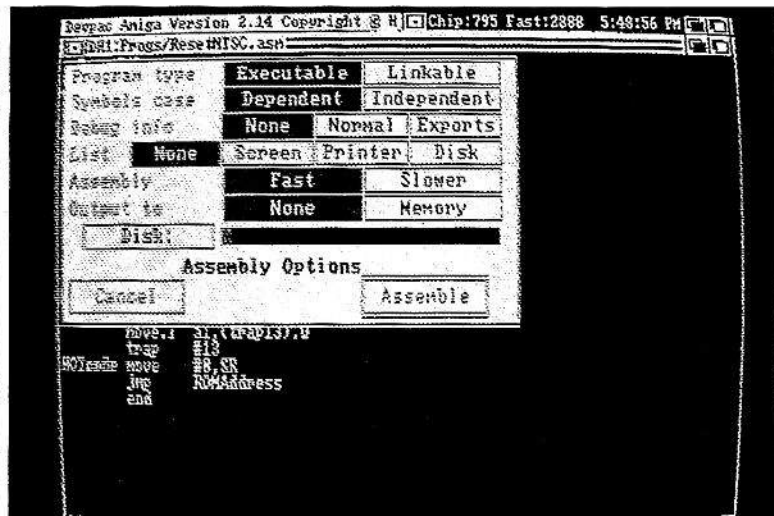
E' quasi perfettamente compatibile con listati scritti per DevPac e CAPE (un assemblatore americano, poco diffuso da noi in Europa) e ne ricalca molte esten-

sioni direttive, oltre ad averne altre ridondanti appunto per migliorare la compatibilità con altri pacchetti non molto standard.

Adapt

Il pacchetto più professionale e costoso (circa **120 dollari**) tra quelli diffusi in Europa, Adapt è tuttora l'unico pacchetto integrato che supporta estesamente tutta la famiglia dei processori 68000 (sino al 68040) e le FPU 68881 e 68882, al contrario degli altri assemblatori che tipicamente accettano solo mnemonici 68000 standard.

E' un macroassemblatore standard Motorola, funzionante senza ambiente integrato, nè editor di linea, in grado di svolgere alcune ottimizzazioni sul codice generato. Il pacchetto è comunque completato da un linker (che accetta librerie di scansione in vario formato, oltre a quello **AmigaDOS** standard, anche **Lattice** ad esempio) e un Program Module Analyzer, ovvero un programma in grado di verificare i tempi di esecuzione delle routine in assembly per consentirne il perfezionamento da parte del programmatore.

Note pratiche

In generale, quando si inizia a lavorare con un assemblatore, si proseguirà con esso per molto tempo, necessario per "costruire" un ambiente ideale alle nostre esigenze e per prendere familiarità con le operazioni manuali da svolgere per lavorare in modo veloce e proficuo.

Per questo motivo è bene partire subito con un buon programma, per non perdere tempo successivamente, quando, superata la fase didattica, si inizierà a lavorare su programmi complessi.

In generale, se non siete dei patiti della digitazione da Shell, è consigliabile lavorare con un assemblatore intuitionizzato, con editor e assembler integrati. Scrivere un programma in linguaggio macchina è un lavoro sempre molto lungo, anche per programmi relativamente semplici.

Inoltre, ogni più piccolo errore causa quasi sempre **Guru Meditation** o dei **crash** completi.

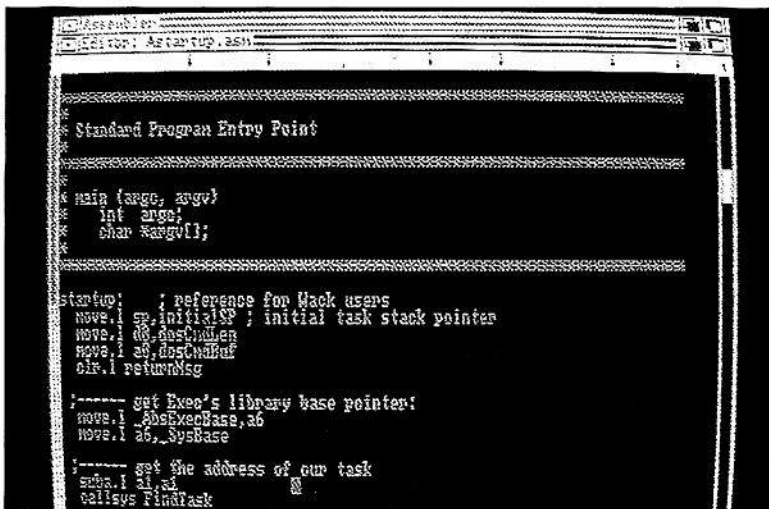
Perciò bisogna prendere l'abitudine di salvare i file su disco o in un sicuro **Ram Disk** recuperabile.

Per le migliori prestazioni, ovviamente, è consigliabile usare un sistema dotato di disco fisso, in modo che dopo un reset si possa rientrare rapidamente in un ambiente completamente configurato senza attendere molto tempo per la lettura di un floppy.

Dipende dallo stile di programmazione e dal tipo di software realizzato, ma ci si può trovare a dovere usare vari file di libreria durante le sessioni di lavoro in Assembler, esattamente come avviene per il linguaggio C.

Anche qui, infatti, si possono usare librerie di linking, file di inclusione e file di libreria con routine scritte precedentemente e salvati in moduli oggetto.

In questi casi diventa veramente utile disporre di almeno un megabyte di Ram per conservare, in Ram recuperabile, i file necessari o, meglio ancora, di un buon Hard Disk.

Prossimamente inizieremo a descrivere sistematicamente, secondo lo stile "a blocchi" già ben noto ai nostri lettori per il "corso" parallelo di Assembler 80X86, tutte le istruzioni del processore 68000, fornendo ovviamente anche alcuni esempi di listati sorgente.

di Luigi Callegari

Come realizzare, in C, un ambiente di lavoro

Chi inizia ad operare in "C", spesso ha difficoltà a personalizzare un dischetto in grado di rappresentare un supporto valido per scrivere programmi, anche impegnativi

Come sanno molti utenti Amiga, i pacchetti dei compilatori C commerciali per Amiga sono due: il **SAS/C** giunto alla versione **5.10** e prodotto dalla SAS (successore del "Lattice C", casa rilevata dalla SAS Incorporated, appunto), e lo **Aztec C** prodotto dalla Manx, giunto attualmente alla versione **5.0d**.

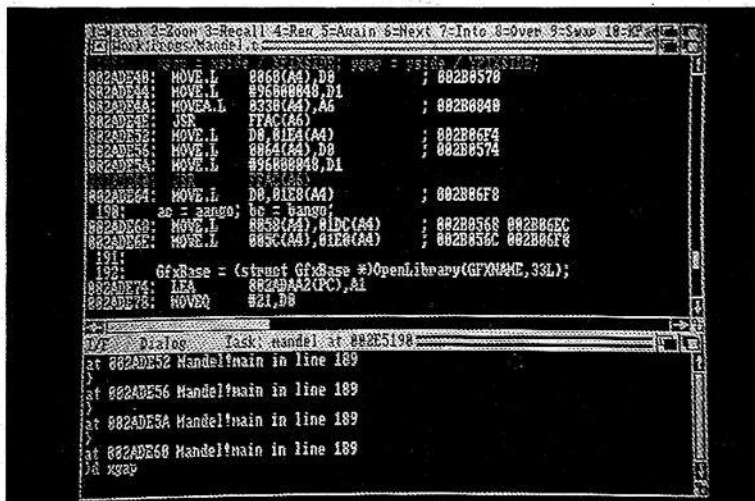
Esistono, effettivamente, anche compilatori di pubblico dominio, ovvero distribuiti *gratuitamente* con la sola richiesta di un *contributo volontario* all'autore da parte di chi li utilizza regolarmente. Il più famoso di questi programmi è **DICE**, scritto da Matthew Dillon, giunto alla versione **2.06B** e rintracciabile, tra l'altro, sul **Fish Disk** numero **466**.

Bisogna rilevare, comunque, che tali programmi sono senza dubbio pregevoli se si considera che sono quasi del tutto gratuiti, ma difficilmente possono essere affidabili, documentati e completi come i pacchetti commerciali. Inoltre i programmi di pubblico dominio sono spesso "abbandonati" dal loro autore, senza quindi che siano corretti i bug noti, nè più ampliati per raggiungere prestazioni soddisfacenti per gli utenti più evoluti e per adeguarsi all'evoluzione del sistema operativo di Amiga, mentre i prodotti commerciali, a fronte di un costo solitamente non trascurabile, sono tipicamente meglio documentati, più completi e continuamente perfezionati nel tempo dalle case produttrici.

Ad ogni buon conto, pensiamo di inserire in uno dei prossimi numeri del nostro **AmiGazzetta** un intero compilatore di pubblico dominio, magari proprio il DICE di Matt Dillon, per consentire anche ai nostri lettori più "squattrinati" di iniziare, con spesa (quasi) nulla, a sperimentare le gioie della programmazione in linguaggio C. Ovviamente ritorneremo a parlare diffusamente di tale ambiente di programmazione in seguito alla pubblicazione.

Questo mese vogliamo parlare, invece, della creazione dell'ambiente di lavoro in linguaggio C con i pacchetti commerciali per Amiga, dando cenni ed indicazioni utili soprattutto a chi **non** ha molta familiarità con la **lingua inglese** usata nei manuali originali.

Parleremo anche di particolari tecnici che difficilmente possono essere ricavati dai manuali stessi, ma sono frutto di studio ed esperienza di una gran mole di lavoro personale.

Del resto crediamo di fare una cosa utile, dal momento che alcune conoscen-

; Installazione SAS/C V5.10 su floppy disk

```
.key to
copy SAS_C_5.1.1:c/copy ram:
path ram:
copy SAS_C_5.1.1:c/delete ram:
copy SAS_C_5.1.1:c/echo ram:
copy SAS_C_5.1.1:c/failat ram:
copy SAS_C_5.1.1:c/makedir ram:
copy SAS_C_5.1.1:c/if ram:
copy SAS_C_5.1.1:c/endif ram:
copy SAS_C_5.1.1:c/assign ram:
echo "Questo script crea un nuovo disco nel drive <to>"
echo "contenente i file di libreria e di inclusione."
echo "Inserisci un disco formattato vuoto in <to>"
ask "Batti Y per continuare, N per terminare"
if warn
assign to: <to>
failat 99
makedir to:lib
makedir to:compacth
failat 20
echo "Inserisci il Disk 2"
copy SAS_C_5.1.2:lib/lc.lib to:lib
copy SAS_C_5.1.2:lib/lcs.lib to:lib
copy SAS_C_5.1.2:lib/lcr.lib to:lib
copy SAS_C_5.1.2:lib/lcsr.lib to:lib
copy SAS_C_5.1.2:lib/lcnb.lib to:lib
copy SAS_C_5.1.2:lib/c.o to:lib
copy SAS_C_5.1.2:lib/catch.o to:lib
copy SAS_C_5.1.2:lib/cres.o to:lib
copy SAS_C_5.1.2:lib/catchres.o to:lib
copy SAS_C_5.1.2:lib/cback.o to:lib
copy SAS_C_5.1.2:lib/lcm#?.lib to:lib quiet
echo "Inserisci il Disk 3"
copy SAS_C_5.1.3:c/lcompact ram:
echo "Inserisci il Disk 4"
assign fp: SAS_C_5.1.4:compiler_headers
assign tp: to:compacth
execute SAS_C_5.1.4:compact
assign fp:
assign tp:
endif
```

Figura 1: installazione SAS/C

ze richieste per ottenere un buon ambiente di lavoro sono tipicamente oscure per chi inizia a lavorare con i pacchetti Amiga, anche se si conoscono già bene le basi teoriche della programmazione in C.

Tali basi sono molto legate alla struttura del sistema operativo di Amiga oltre che alle caratteristiche intrinseche e peculiari dei compilatori stessi.

Floppy e Hard Disk

Chiariamo subito che per lavorare in termini soddisfacenti con un compilatore C è praticamente indispensabile disporre di un **hard disk**.

Certo, è possibile anche lavorare con due o tre floppy disk, disponendo di almeno **un megabyte** di Ram e usando accorgimenti come il copiare in **Ram Disk** parte degli eseguibili, magari proteggendoli dal reset tramite **VD0:** od un altro Ram Disk recuperabile.

Ma, effettivamente, i tempi di elaborazione e l'elasticità delle operazioni di redazione, compilazione, linking, verifica e debugging tramite appositi programmi dei propri lavori diventano decisamente molto lunghi, già per programmi quasi banali.

Chi desidera usare con profitto il proprio Amiga, 500 o 2000 (ma anche altri modelli di personal computer, come gli Ms - Dos, in effetti), nel campo della programmazione o semiprofessionale deve valutare seriamente l'acquisto di un HD, tra i tanti modelli disponibili oramai in Italia e elencati nelle pagine delle inserzioni pubblicitarie.

I pacchetti commerciali di programmazione in linguaggio C per Amiga, comunque, possono installare un sistema operativo sia su di un sistema dotato di almeno due floppy disk drive, sia su hard disk, tramite appositi file batch.

In **figura 1** riportiamo, a titolo di esempio, la traduzione del file di installazione del **SAS/C V5.10** in un sistema a floppy disk.

Installando un sistema su floppy disk bisogna, tra l'altro, sacrificare alcune "librerie" e standardizzare il tipo dei programmi che si realizzano con un certo "gruppo" di dischetti. Su disco fisso, ovviamente, si può installare il sistema completo per qualunque esigenza.

Installazione Aztec C

Ambedue i pacchetti commerciali per Amiga nelle versioni più recenti sono dotati di programmi di installazione, pertanto la procedura per creare gli ambienti di lavoro è oltremodo semplificata.

Nel caso di Aztec C, inserendo al **boot-strap** (inizializzazione dopo l'accensione della macchina, quando appare la manina col disco del Workbench, insomma) il **primo** dischetto dei **cinque** forniti, appare una serie di richieste per l'installazione. Dunque, si badi bene, si lavora comunque, anche in un sistema privo di disco rigido, con copie dei dischetti originali, i cui contenuti, parziali rispetto ai dischi originali acquistati, sono fissati dal programma di installazione.

Nel caso di Aztec C, occorrono almeno due dischetti da 3,5" già formattati per consentire la creazione dell'ambiente di

Le librerie di SAS/C

Ecco l'elenco delle librerie di linking fornite con il **Lattice** SAS/C V5.10. Da notare che le suddivisioni delle librerie sono dovute sia alla dimensione degli interi, sia alla previsione di funzioni matematiche, sia all'uso di parametri in registri, sia all'indirizzamento relativo od assoluto.

```
lc.lib       Standard, con interi a 32 bit
lcnb.lib     Standard, con indirizzamento assoluto
lcr.lib      Standard, con parametri registrati
lcs.lib      Standard, con interi a 16 bit
lcsr.lib     Standard (INT=16 bit), moduli residenti

lcm.lib      Matematica IEEE doppia precisione
lcm881.lib   Matematica per FPU 68881
lcmffp.lib   Matematica Fast Floating Point
lcmieee.lib  Matematica IEEE per librerie residenti
lcmr.lib     Matematica IEEE con parametri registrati
lcms.lib     Matematica IEEE con interi a sedici bit
```

Le librerie SAS/c

lavoro. In pratica, la procedura è del tutto automatica e prevede semplicemente che l'utente inserisca, a turno, i due dischetti.

Al termine, appare una finestra intitolata **Aztec 5.0 Floppy Install**. Da qui, tramite gadget, si può selezionare il tipo di ambiente di lavoro desiderato, secondo criteri che descriveremo tra poco a beneficio dei meno esperti: grandezza degli interi, modelli di memoria e librerie matematiche.

Questi concetti sono comuni ad ambedue i compilatori commerciali per Amiga e strettamente dipendenti dal tipo di computer usato, quindi non trasportabili in altri ambienti di lavoro e li approfondiremo tra poche righe.

Installazione SAS/C

Con il Lattice/SAS C, inserendo al bootstrap il **primo** dei **sei** dischetti AmigaDOS viene eseguito un file di startup che richiederà l'inserimento a turno di alcuni dischetti.

Infatti, a differenza di Aztec C, il SAS/C crea al boot un sistema già utilizzabile, con tutti gli **Assign** necessari al funzionamento del compilatore. In pratica, per avere un sistema SAS/C su floppy disk già utilizzabile è sufficiente copiare i dischetti di backup, teoricamente.

In effetti, come suggerisce il manuale, è consigliabile eseguire uno dei due file batch forniti nella directory **S** del primo dischetto: **`install_floppy`** (per sistemi a dischetto con 512k), **`install_floppy_1m`** (per sistemi a dischetto con 1 Mb) o **`Install_hd`** (per sistemi dotati di disco fisso).

Essendo SAS/C più goloso di memoria durante il funzionamento, nonchè più mastodontico per dimensione dei file eseguibili (**LC**, **LC1**, **LC2**, **BLink**) rispetto a quelli di Aztec (**CC**, **AS** e **LN**), è particolarmente consigliabile disporre di un disco fisso e di almeno un megabyte di memoria per lavorare in modo soddisfacente.

Abbiamo riportato in figura il listato tradotto del file batch di installazione del sistema su floppy, che può essere di utile riferimento a tutti i possessori di SAS/C o di versioni piuttosto vecchie di Lattice C (che non avevano questi file di installazione).

Grandezza degli interi

Usando le librerie di linking e le opzioni di compilazione "di default", i programmi compilati con le versioni 5.10 di SAS/C e 5.0 di Aztec C usano 32 bit per conservare i numeri interi.

Dal momento che tutte le funzioni delle librerie di Amiga si aspettano di ricevere valori di input a 32 bit nei registri del 68000, usando INTeri con questa dimensione ci si assicura che nelle chiamate a tali funzioni non si avranno errori. Ricordiamo, infatti, che una linea in C del tipo...

```
Draw (RP, 5L, 6);
```

...chiama la funzione **Draw()** della libreria Graphics di Amiga, passandole come tre valori di input un **puntatore**, il numero **5** in formato Long (come da noi esplicitamente indicato con la "**L**") ovvero a 32 bit, e il numero **6.** Dal momento che quest'ultimo non è seguito da una specifica di dimensione, il compilatore passerà

```
Lattice Screen Editor
{
    printf("\nErrore %d: %s\n",n,s);
    exit( (LONG)EX_ERROR );
}

/* Alloca memoria per un nuovo nodo */
struct node *palloc( void )
{
    return( (struct nodo *) malloc(sizeof(struct nodo)));
}

/* Alloca memoria e salva una parola nuova */
char *savestr( char *s )
{
    register char *p;

    p = (char *) malloc( strlen( s ));
    if ( p == NULL ) return( p );
    else             return( (char *)strcpy( p, s) );
}

/* Stampa ricorsivamente l'albero */
void displyalbero( struct nodo *p )
{
    if ( p != NULL )
    {
        displyalbero( p -> min );
LINE: 81   FILE: Vlexto             INSERT (1)
Text entry mode. Press F1 for LSE help.
```

Librerie Aztec C

Ecco l'elenco delle librerie di linking per Aztec C V5.0. La suddivisione dipende dalla dimensione degli interi, dalle librerie matematiche e dai modelli di gestione della memoria, ovvero: Small Code (SC) e Small Data (SD) o Large Code (LC) e Large Data (LD).

```
c.lib       Standard, INT=32 bit, SC SD
c16.lib     Standard, INT=16 bit, SC SD
cl.lib      Standard, INT=32 bit, LC LD
cl16.lib    Standard, INT=16 bit, LC LD

m.lib       Matematica Manx IEEE, INT=32 bit, SC SD
ma.lib      Matematica Amiga IEEE, INT=32 bit, SC SD
mf.lib      Matematica Mototorola FFP, INT=32 bit, SC SD
m8.lib      Matematica FPU 68881, INT=32 bit, SC SD

ml.lib      Matematica Manx IEEE, INT=32 bit, LC LD
mal.lib     Matematica Amiga IEEE, INT=32 bit, LC LD
mfl.lib     Matematica Motorola FFP, INT=32 bit, LC LD
m8l.lib     Matematica FPU 68881, INT=32 bit, LC LD

m16.lib     Matematica Manx IEEE, INT=16 bit, SC SD
ma16.lib    Matematica Amiga IEEE, INT=16 bit, SC SD
mf16.lib    Matematica Motorola FFP, INT=16 bit, SC SD
m816.lib    Matematica FPU 68881, INT=16 bit, SC SD

ml16.lib    Matematica Manx IEEE, INT=16 bit, LC LD
mal16.lib   Matematica Amiga IEEE, INT=16 bit, LC LD
mfl16.lib   Matematica Motorola FFP, INT=16 bit, LC LD
m8l16.lib   Matematica FPU 68881, INT=16 bit, LC LD
```

Una serie di librerie personali di Aztec C è quella che contiene delle funzioni di gestione dello schermo ANSI. Si tratta di sole quattro librerie:

```
s.lib       Schermo, INT=32 bit, SC SD
s16.lib     Schermo, INT=16 bit, SC SD
sl.lib      Schermo, INT=32 bit, SC SD
sl16.lib    Schermo, INT=16 bit, LC LD
```

Le librerie Aztec C

un INTero normale, che se fosse a 16 bit provocherebbe probabilmente, come minimo, un malfunzionamento della Draw().

Usando invece interi a 32 bit, non sarà necessario esplicitare di portare a Long, ovvero a 32 bit, tutti i valori da passare a funzioni, come quelle delle librerie di Amiga, che richiedono parametri di queste dimensioni.

Si noti che la spiegazione di questo problema è legato alla gestione dei formati dei dati da parte del processore 68000, argomento che esula le competenze di questa serie di articoli e che sarà invece trattato nella serie di articoli sulla programmazione di Amiga in linguaggio Assembler.

Sezioni di un programma

Prima di addentrarci nella spiegazione della differenza materiale tra le caratteristiche dei modelli di memoria, occorre chiarire come è composto un programma compilato in C.

In AmigaDOS, un programma scritto in C è costituito da quattro sezioni tipiche chiamate **code, data, stack** e **heap**. Con la prima si indica quella porzione di codice eseguibile che contiene il codice assembly propriamente detto, che viene, cioè, fatto eseguire dal processore. La sezione "data" è composta invece dai dati del programma: frasi, costanti numeriche od alfanumeriche, eccetera. Questa, effettivamente, è suddivisibile in altre due sezioni, quella dei dati inizializzati e quella dei dati non inizializzati, ma tale distinzione è importante solo a livelli piuttosto avanzati di programmazione, quando ci si trova a dovere operare con **hunk** e programmi residenti, ad esempio, perciò per ora ne tralasciamo le spiegazioni.

Lo "stack" è quella parte di memoria, riservata al programma dal sistema operativo di Amiga (Exec) al momento della sua esecuzione (da Shell o Workbench), dove vengono conservate le variabili automatiche e temporanee e le informazioni di controllo dell'esecuzione del programma, usate dal processore.

Infine, lo "heap" è l'area di memoria dove vengono allocati e deallocati **dinamicamente** i buffer, ovvero le aree di lavoro del programma. Per gestione "dinamica" di un'area di memoria si intende la sua gestione **durante** l'esecuzione stessa del programma, quindi non quanto prestabilito dal programmatore con allocazioni fisse al momento della scrittura del sorgente, ma tipicamente tramite chiamate a funzioni di libreria tipo **malloc()** o **getmem()**.

Modelli di memoria

Il modello di memoria usato da un programma determina il modo in cui il codice eseguito dal processore fa riferimento al codice e ai dati.

Il che, indirettamente, influenza pure la dimensione del codice finale, la velocità di esecuzione del programma e la massima quantità di dati e di codice eseguibile che possono costituire il programma stesso.

La differenza fondamentale tra un programma che usa il modello di gestione della memoria **large data** rispetto ad uno che usa il modello **small data** è nelle istruzioni usate nel codice eseguibile (inserite dal linker prelevandole dalle opportune librerie di scansione, vedi riquadri) per fare riferimento al segmento di programma che contiene i dati, come detto prima.

Un programma generato usando il modello "small data" accede ai dati contenuti nella pertinente sezione usando istruzioni del processore "position independent", ovvero che fanno riferimento ai dati con uno scostamento (**offset**) rispetto ad una posizione data, solitamente contenuta in un apposito registro indice del processore 68000.

Invece, il modello "large data" usa istruzioni di riferimento assoluto in memoria.

Le differenze che derivano tra i due modelli sono disparate, come preannunziato. Innanzitutto non vi è limite teorico (se non legato alla Ram della macchina) alla quantità di dati globali e statici che un programma di tipo "large data" può gestire, mentre un codice "small data" può avere al massimo 64K di dati statici e globali.

Il DOS impiega più tempo a caricare il segmento di codice appartenente ad un programma large data, dato che per la particolare struttura di Exec (multitasking, ricordiamo), l'indirizzo effettivo di caricamento in memoria di un programma non è mai noto sinchè appunto non viene letto dal disco; dopo è compito di AmigaDOS calcolare e correggere tutti i riferimenti "assoluti" del file eseguibile caricato a che indichino effettivamente delle locazioni di memoria. Programmi small data che usano riferimenti indiretti non hanno bisogno di queste correzioni, dato che la loro struttura è già "rilocabile" quando viene caricato da disco.

Un programma large data è tipicamente più lungo, dato che usa per i riferimenti alla sezione dei dati delle istruzioni con indirizzamento a 32 bit (in grado, appunto, di indicare qualunque locazione nella mappa di memoria di Amiga), mentre un programma small data usa un campo a 16 bit (offset) per riferimenti indiretti.

Un programma che usa il modello large data è tipicamente un po' più lento in esecuzione, poichè il processore necessita di più tempo per eseguire una istruzione di riferimento assoluto in memoria rispetto a quando può usare uno scostamento rispetto ad un registro indice, come avviene per il modello small data.

Infine, un programma small data usa un registro, tipicamente **A4**, per contenenere l'**indirizzo di base** del blocco di dati, a cui fare riferimento per indirizzare indirettamente (con un offset) i dati, mentre un large data non "spreca" questo registro, che è quindi utilizzabile per altri scopi.

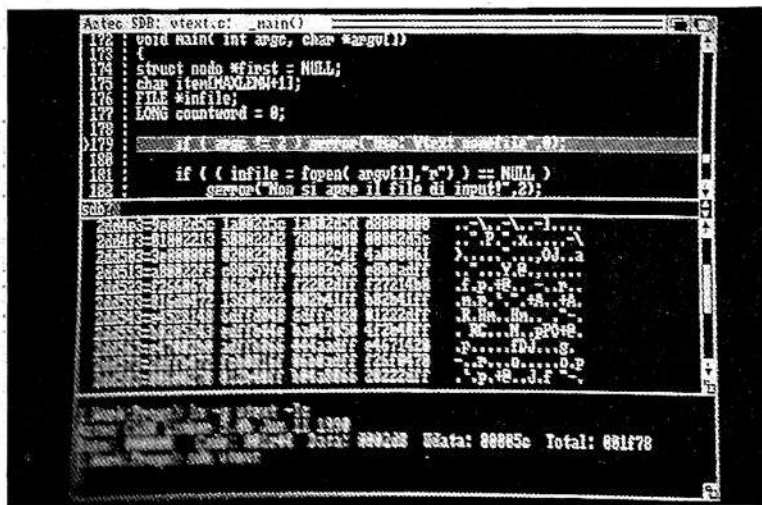
Large Code o Small Code

Un discorso analogo a quello appena visto per i dati può essere fatto per il tipo di codice generato dal compiler.

Un programma compilato secondo il modello "small code" usa sempre delle istruzioni indipendenti dalla posizione per fare riferimento a sezioni di codice eseguibile (ad esempio, per le istruzioni di salto), mentre un programma "large code" usa sempre delle istruzioni assolute. Le conseguenze sono che ogni istruzione che fa riferimento ad una porzione del programma (ad esempio, una istruzione di salto condizionato) deve necessariamente avere l'indirizzo a cui fa riferimento distante al massimo 32K (prima o dopo) in memoria. In effetti, il linker è solitamente in grado di generare comunque una istruzione corretta, usando un "trucco": crea nella sezione dati una tabella di salti assoluti (jump table) a cui fa riferimento sostituendo l'istruzione di salto indiretto "insufficiente" con una di salto (o chiamata) a 32 bit (assoluta) nella locazione adeguata della jump table. Tutto ciò, se consente al programma di funzionare comunque, superando il limite di 32K delle istruzioni indirette del processore 68000, introduce però rallentamenti nell'esecuzione e dati supplementari, che allungano la sezione dati.

Un programma small code, per accedere alla jump table, usa un registro indice supplementare, che viene fatto puntare alla locazione centrale della jump table, i cui elementi quindi vengono indicati tramite il solito indirizzamento indiretto. Se il programma usa il modello small data, viene usato questo stesso registro indice sia per accedere alla jump table, sia per accedere al segmento dei dati. Per un programma large code, questo registro non è ovviamente necessario per fare riferimento a locazioni nel segmento di codice.

Un segmento di tipo "code" può effettivamente contenere anche dei dati. Una istruzione di un programma large code può accedere ai dati localizzati ovunque nel segmento di codice, perchè accede al segmento dei dati con istruzioni assolute, usando indirizzamenti a 32 bit. Invece, l'istruzione di un programma small code userà un riferimento indiretto a 16 bit, quindi distante al massimo 32K dalla istruzione.

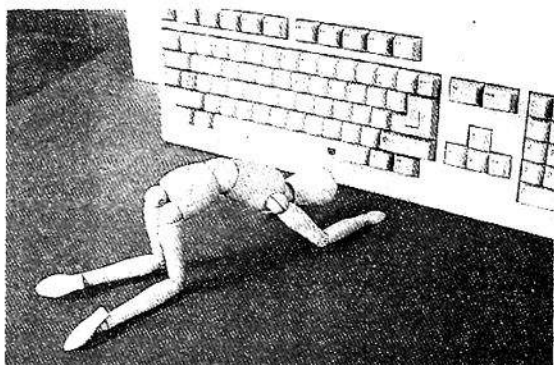
Un programma large code viene caricato più lentamente anche perché devono essere corretti da AmigaDOS tutti i riferimenti assoluti in modo che indichino indirizzi effettivi (noti solo al momento del caricamento, essendo la memoria di Amiga gestita in modo molto flessibile, essendo un computer multitasking) e viene eseguito più lentamente, dato che le istruzioni assolute richiedono più tempo per essere eseguite rispetto a quelle "PC-relative" tipiche del modello small code.

Infine, un programma generato in modello large code è tipicamente un po' più lungo di uno che usa il modello small code, dato che le istruzioni di riferimento assoluto sono più lunghe di quelle "PC-relative".

La matematica

Il terzo ed ultimo argomento da trattare per capire come si configura durante l'installazione un compilatore per Amiga, nonchè come si lavora in seguito, è legato alla presenza di varie librerie matematiche in Amiga.

Bisogna infatti rammentare che per svolgere le funzioni matematiche in virgola mobile, occorre fornire al linker delle librerie di scansione supplementare, ove reperire quelle funzioni matematiche da inserire nel codice oggetto.

Normalmente, si assume che un programma in C non abbia bisogno di gestire numeri in virgola mobile, ma semplicemente numeri interi.

In pratica, il programmatore, limitandosi all'uso dei vari Long, Int e Short e tralasciando i Float e i Double, può così ottenere programmi molto più veloci e compatti di quando deve usare librerie matematiche per manipolare numeri in virgola mobile.

Una prima libreria matematica in virgola mobile è la cosiddetta **Motorola FFP**, ovvero **F**ast **F**loating **P**oint, che usa routine in singola precisione per eseguire i calcoli matematici. I programmi compilati con le apposite opzioni e linkati con le corrette librerie (vedere riquadri) per la matematica FFP, saranno più lenti di quelli utilizzanti solo matematica intera ed un po' più veloci di quelli utilizzanti la matematica in doppia precisione.

La libreria probabilmente più usata e consigliabile è invece la **Amiga IEEE**.

In pratica, il programma compilato può usare le routine comprese nei file di libreria inseriti dalla Commodore nella directory LIBS del disco di sistema, ovvero **mathieeedoubbas.library** e **mathieeedoubtrans.library**. Il programma, al momento della esecuzione, dovrà potere caricare dal disco di sistema (directory LIBS) tali librerie per eseguire, rispettivamente, le funzioni matematiche e scientifiche in doppia precisione e le funzioni trascendenti.

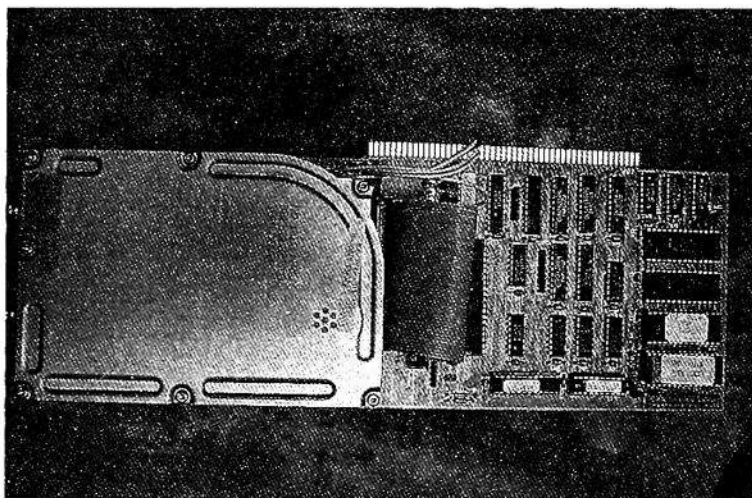
I compilatori Amiga dispongono anche di un set di librerie personali, scritti dalla Lattice e dalla Aztec, che svolgono i calcoli in doppia precisione standard IEEE.

In generale, se tali librerie possono produrre risultati diversi nella qualità del codice ottenuto (solitamente un po' più lungo ma più preciso rispetto a quando si sfruttano le librerie Commodore Amiga), sono sconsigliabili perchè se il computer ospite del nostro eseguibile dispone di un coprocessore matematico, questo non verrà utilizzato.

Difatti, mentre le funzioni matematiche delle librerie Commodore Amiga, quelle in LIBS per intenderci, sono in grado di rilevare la presenza ed usare automaticamente l'eventuale FPU (Floating Point Unit), con un enorme aumento della velocità di esecuzione in programmi matematici, le librerie Aztec e Lattice funzionano sempre e comunque utilizzando il processore 68000.

Ricordiamo che, oggigiorno, non sono rari gli utilizzatori di FPU per Amiga: tutti i possessori di Amiga 3000 e di schede acceleratrici per Amiga 2000 (e buona parte di quelle per A-500) montano di serie una FPU.

Nel caso la velocità di elaborazione sia estremamente importante per un programma matematico, si può anche produrre una versione speciale del programma che può girare SOLO su computer dotati di FPU MC68881 o MC68882, sfruttando appunto la quarta libreria di compilazione matematica in dotazione ai compilatori C per Amiga. In questo caso il programma finale non userà le librerie LIBS, ma piloterà direttamente il coprocessore matematico, che deve essere presente (su di un Amiga "normale" il programma genera altrimenti dei Guru), ottenendo la massima velocità possibile, superiore anche di dieci volte tipicamente a quella di un Amiga con solo 68000.

Il mese prossimo, sulle pagine di Computer Club, riprenderemo la programmazione di Amiga in C, addentrandoci nelle librerie di funzioni esclusive e potentissime.

Disco delle mie brame...
Leggendo Postamiga del n. 77 ho capito come far riconoscere ad un programma il nome o la data di un dischetto, ma vorrei sapere se è possibile anche stabilire se un disco è "write protected" oppure no, e quanto spazio libero ha ancora. Si usa la stessa libreria?
(Giacomo Zampi - Roma)

La libreria non può che essere la stessa (**Dos library**), ma va sfruttata una diversa funzione.

Per i "distratti", si ricorda che il termine **Libreria** allude ad una serie di routine raggruppate in un unico contesto: ognuna di queste routine costituisce una **Funzione**.

Il listato cui la missiva fa riferimento sfruttava la funzione **EXAMINE** della **Dos.library**, la quale **non** può fornire il tipo di informazione che interessa il lettore.

Allo scopo, è necessario ricorrere ad un'altra funzione, di nome **Info**, che a grandi linee va adoperata in maniera simile a Examine. Ovvero: una volta invocata, questa restituisce una struttura di dati (di nome **InfoData**) contenente una serie di informazioni, tra le quali quelle che interessano.

Per la precisione, la struttura sarà composta da **9** long word, per un totale di **36** byte (1 long word = 4 byte), ognuna delle quali rappresenta un tipo di informazione. Per leggerle da Basic, oltre a conoscerne il significato (che vedremo tra breve), sarà quindi necessario:

☞ 1) Aprire la libreria del dos, dichiarando che si intende usare la funzione Info.

☞ 2) Predisporre un'area di memoria riservata, nel nostro caso di almeno 36 byte, che dovrà contenere la struttura dati.

☞ 3) Richiamare la funzione Info.

☞ 4) Leggere i dati che sono stati inseriti nella struttura.

☞ 5) Chiudere la libreria.

Questo, come vedremo meglio in pratica, a grandi linee, visto che, per allocare la memoria atta a contenere la struttura, si rende obbligatorio aprire **anche** la libreria **Exec**, ed usare la sua funzione **Allocmem**, sulla quale ci siamo spesso dilungati. Ma procediamo a piccoli passi, in modo da rendere chiara la cosa anche ai meno esperti.

La struttura InfoData, rapportata all'uso "basico", può essere così descritta (i nomi sono quelli standard che i più evoluti potranno rintracciare negli **Include** del loro compilatore o assemblatore):

```
base +0  = NumSoftError
base +4  = UnitNumber
base +8  = DiskState
base +12 = NumBlocks
base +16 = NumBlockUsed
base +20 = BytesPerBlock
base +24 = DiskType
base +28 = VolumeNode
base +32 = InUse
```

L'interpretazione è piuttosto semplice: **base** corrisponde, in pratica, all'indirizzo di memoria a partire dal quale è memorizzata la struttura che, essendo composta da sole long word, avrà i suoi dati disposti con cadenza di **4** byte. Facile (o quasi) ricavare dalla struttura le informazioni sullo spazio disponibile in un certo

```
PRINT "ROUTINE PER CONOSCERE LO STATO"
PRINT "DEI DISCHI ACCESSIBILI AL SISTEMA"

DECLARE FUNCTION info& LIBRARY
DECLARE FUNCTION lock& LIBRARY
DECLARE FUNCTION allocmem& LIBRARY
LIBRARY "dos.library"
LIBRARY "exec.library"
ON BREAK GOSUB nostop:BREAK ON
'-------------------------------------
flag=0
start:
CLS:PRINT:PRINT"0 = Df0:":PRINT"1 = Df1:"
PRINT"2 = Dh0:":PRINT "3 = Input nome"
PRINT "4 = fine":PRINT:PRINT"SCEGLI"
loop:
a$=INKEY$:IF a$="" THEN loop
a=ASC(a$):IF a<<48 OR a>>52 THEN loop
IF a$="4" THEN
  IF flag=1 GOTO fine1
  IF flag=0 GOTO fine3
 END IF
IF a$="2" THEN
  drive$="DH0:"
ELSE
  drive$="df"+a$+":"
END IF
IF a$="3" THEN
  PRINT "Nome disco (con eventuali ";
  PRINT "due punti finali)"
  INPUT drive$
END IF
access%=-2:mem&=40
LL&=lock&(SADD(drive$+CHR$(0)),access%)
IF LL&=0 THEN
  PRINT "Lock non ottenuto":GOTO fine3
END IF
buffer&=allocmem&(mem&,65536&)
IF buffer&=0 THEN
  PRINT "Fallita allocazione!":GOTO fine2
END IF
leggi%=info&(LL&,buffer&)
IF leggi%=0 THEN
  PRINT "Info non ottenuto!":GOTO fine1
END IF
'-------------------------------------
flag=1:COLOR 3,0:PRINT
PRINT "Unita' Numero     -";PEEKL(buffer&+4)
PRINT "Tipo di disco     - ";
FOR tipdisk=0 TO 2
 PRINT CHR$(PEEK(buffer&+24+tipdisk));
NEXT:PRINT
Diskstate&=PEEKL(buffer&+8)
PRINT "Stato disco       - ";
IF Diskstate&=80 THEN
 PRINT "Protetto in scrittura"
ELSEIF Diskstate&=82 THEN
 PRINT "Agibile in scrittura"
ELSEIF Diskstate&=81 THEN
 PRINT "In fase di validazione"
END IF
totali&=PEEKL(buffer&+12)
usati&=PEEKL(buffer&+16)
bytes&=PEEKL(buffer&+20)
bfree&=(totali&*bytes&)-(usati&*bytes&)
PRINT "Blocchi totali   -";totali&
PRINT "Blocchi usati    -";usati&
PRINT "Byte per blocco  -";bytes&
PRINT "Byte liberi      -";bfree&
PRINT:PRINT:COLOR 1,0
'-------------------------------------
REM PRINT :PRINT "PREMI UN TASTO"
REM loop2:
REM a$=INKEY$:IF a$="" THEN loop2
REM CALL unlock(LL&):GOTO start
'-------------------------------------
fine1:
CALL freemem(buffer&,mem&)
fine2:
CALL unlock(LL&)
fine3:
LIBRARY CLOSE:END
nostop:
PRINT"uscita solo da menu!!":RETURN
```

disco (vedremo dopo come indicare al sistema il disco da trattare): la voce **Numblocks** ci dirà espressamente quanti blocchi sono globalmente disponibili sul disco, **NumBlockUsed** quanti ne sono stati utilizzati, e **BytesPerBlock** il numero di byte utilizzabili in ogni singolo blocco. Qualche banale operazione aritmetica, e conoscere il numero di byte liberi sarà un gioco.

Per conoscere lo stato del floppy, andrà invece interrogata la long word posta a "distanza" Base+ 8, ovvero **DiskState**. Questa conterrà un valore decimale **80** se il disco è protetto in scrittura, **82** se è agibile, **81** se è ancora in fase di validazione. Semplice, no?

Altri dati interessanti possono essere **UnitNumber**, che fornisce il numero di unità fisica nel quale è inserito il floppy (**0** per df0:, **1** per df1:, eccetera, ma occhio: 0 anche per **Hd0:**, 1 anche per **Hd1:**, e così via), oppure **DiskType**, peraltro poco utile da basic: il valore negativo di **-1** indicherà "disco non presente", ma saranno i requester di sistema ad intervenire prima.

Dopo tanta teoria, un po' di pratica.

In queste pagine è presente un listato Basic che applica quanto finora descritto, in maniera sufficientemente lineare. Senza stare a descrivere tutto, accenniamo solo a qualche passaggio che potrebbe risultare ostico.

Come si accennava prima, è necessario fornire al sistema il nome del disco (o della periferica, per esempio Df1:) da esaminare.

Per far ciò, è necessario ricorrere a un'altra funzione

della Dos Library, **Lock**, già ampiamente descritta sul numero **77** (ed in altra *Postamiga*). Prima di richiamare la funzione Info, il programma "chiede" un Lock sul nome del disco fornito in input. In pratica, significa usare come parametro per Lock l'indirizzo (ottenuto con **Sadd**) della relativa variabile (nel listato: **Drive$**).

Si noti, come già detto sul n. 77, l'uso di un flag (variabile **Flag** nel listato) per sapere se il programma ha "girato" almeno una volta: necessario per chiudere il Lock o le librerie di conseguenza, evitando visite di eventuali Guru.

Le informazioni dalla struttura, molto banalmente, sono ottenute con semplici **Peekl** (si ricordi che i dati sono long word), e non ci si stupisca per apparenti incongruenze: la funzione Info restituisce l'effettivo ammontare in byte dei blocchi, come pure il loro numero globale, senza valutare in questo calcolo i blocchi o gli header adoperati dal sistema. Quindi, per un File System normale, un blocco disporrà di **488** byte per i dati, non **512** come ci si potrebbe aspettare.

Allo stesso modo, il numero globale di blocchi non conterrà quelli usati per il boot.

Chi possiede un hard disk formattato in **Fast File System**, noterà invece come il programma fornirà il valore 512 byte per blocco, proprio perché l'FFS utilizza per intero lo spazio disponibile in ogni blocco.

Si notino, nel listato, le 4 righe disattive per la presenza di **Rem**. Se si eliminano (i soli Rem), il programma riprende la sua esecuzione senza arrestarsi. Le si sono inserite solo per evidenziare un obbligo: dopo ogni suo utilizzo, è indispensabile "Unlockare" quanto prima si era Lockato. Se non lo si fa, si riscontrerebbe il classico fenomeno (per esempio) della mancata scomparsa in Workbench dell'icona del disco esaminato, che continuerebbe ad occupare inutilmente delle risorse di sistema, e questo per **tutti** i dischi esaminati!

Chi è ai primi passi non dimentichi, inoltre, che il listato non può funzionare da solo, semplicemente copiandolo e mandandolo in esecuzione: occorre siano presenti nella stessa **directory corrente** (eventualmente modificabile da basic con **Chdir**) i file **Dos.bmap** e **Exec.bmap**, rintracciabili nella directory **Basicdemos** del disco **Extras**.

In alternativa, si possono modificare le istruzioni **Library** in testa al listato, in modo che precisino la directory ove rintracciarli; per esempio:

```
LIBRARY "df1:basicdemos/
dos.library"
```

L'argomento, lo si sarà notato, non è proprio semplicissimo, ma... c'è di peggio.

Ti telefono un file

Mi hanno detto che esiste un modo per mandare un file in un'altra città con il modem, collegandosi con una bbs della mia città. E' vero? Sarebbe un bel risparmio sulla bolletta...
(Pietro Giotti - Roma)

La domanda non è molto chiara, ma probabilmente si riferisce ad un metodo, talvolta usato nell'ambito di reti nazionali, per inviare/ricevere brevi files codificati in modo tale da poter essere manipolati come se si trattasse di testi.

Come ben sanno i frequentatori telematici della rete amatoriale **Fidonet**, esistono infatti due programmini per Amiga, **Uuencode** e **Uudecode**, in grado di effettuare questa conversione nei due sensi: Uuencode trasforma un file binario in modo che tutti i suoi byte possano essere espressi da un carattere ASCII, mentre Uudecode effettua l'operazione inversa.

In pratica, sarà sufficiente *"uuencodare"* un file, inserire il file ASCII (ovviamente incomprensibile) in un proprio messaggio, e indirizzare quest'ultimo al destinatario.

Il ricevente preleverà il file di pseudotesto, lo "uudecoderà", e si ritroverà il file pronto per l'uso.

Questa tecnica, adatta soprattutto per brevi file (magari preventivamente compressi), può in effetti essere adoperata per invii da una città all'altra collegandosi via modem ad una bbs aderente a una rete nazionale, ma ad una condizione: che il sysop titolare sia consenziente.

E' vero che questa tecnica viene talvolta adoperata, ma principalmente per file di interesse pubblico più che per invii "personali".

Se mittente e destinatario si trovano infatti nella stessa città, la cosa non comporta oneri particolari, ma nel caso di... interurbane, sarà il sysop (e non solo uno) a sobbarcarsi delle spese telefoniche. Molto, insomma, dipende dal suo buon cuore...

Repetita iuvant

La mia stampante funziona perfettamente, tranne cha da Basic. Non riesco infatti ad abilitare alcune sue modalità come il grassetto (ma lo stesso dicasi per il colore ed altre prestazioni), per il quale il manuale dice di inviare un codice di Escape+E con Lprint Chr$(27); Chr$(69).
(Italo D'orazio - Roma)
(Ivan Paiano - Lecce)

Il problema, nonostante sia stato già affrontato su Postamiga, ci viene da tempo riproposto da più lettori, per cui rispondiamo brevemente, rimandando per maggiori dettagli ed esempi ai numeri **65** e **75** della rivista.

Per inviare il codice di **Escape** alla stampante, l'istruzione **Lprint** ***non*** va bene.

E' invece opportuno aprire un file in output direttamente rivolto alla porta parallela, e adoperare l'istruzione Basic **Print#** per inviare il **Chr$(27)**, seguito dai codici specifici della stampante, a seconda del compito che si intende farle svolgere. L'istruzione indicata dal lettore, andrà quindi così strutturata...

```
Open "par:" For Output As 1
Print# 1,Chr$(27);Chr$(69);
Close #1
```

Speciale Principianti

Primi dubbi

Sono un principiante possessore di Amiga 500 e vorrei porvi alcune domande.
1) Quando analizzo i file di un dischetto, alcuni mi danno il seguente errore: File Contains Binary. Come mai?
2) Come si fa a decodificare musiche e schermate grafiche in modo che si possano attivare da Shell?
(Gabriele Felici - Livorno)

I file presenti in un dischetto possono essere di più tipi.

Tutti, da un punto di vista fisico, sono composti da sequenze numeriche, che però possono assumere un diverso significato per il computer. Generalizzando alquanto per non confondere ulteriormente le idee, ci limiteremo qui a dire che i codici numerici contenuti in alcuni di essi rappresentano dei testi, mentre altri memorizzano programmi da eseguire.

Il tipo di errore verificatosi, nonostante la genericità della domanda, fa supporre che l'"analisi" del file sia stata tentata con un lettore di testi, ovvero con un programma adatto per leggere il contenuto di un file ASCII (un codice di numeri che rappresentano, ognuno, un carattere alfanumerico).

Essendo il programma di lettura specializzato in questo compito, evidentemente si è "accorto" che il file da esaminare, o meglio (secondo lui) di cui far leggere il contenuto, non era un file di testo, ma un file "binario", termine con il quale si definisce una qualunque altra sequenza numerica memorizzata in un file (non si dimentichi che il computer "ragiona" proprio in notazione binaria). La terminologia fin qui adoperata è necessariamente restrittiva, ma giusto per farsi capire anche da un neofita...

Quanto alla seconda domanda: non è necessario "decodificare" un file contenente musica o schermate grafiche. E' sufficiente adoperare programmi adatti allo scopo, peraltro molto diffusi. Per il suono, per esempio, esiste tutta una serie di **Play** (di solito è questo il loro nome) per cui è sufficiente digitare da **Shell Play Nomefile** per ascoltare il brano. Il programma Play (o similare, a seconda dello standard del file), in mancanza di esperienza, lo si può copiare nella directoy **C** del disco Workbench.

Lo stesso dicasi per visualizzare una schermata grafica, solo che andrà usato un qualche programma (ne esistono una infinità) come **Show**, dall'uso analogo a quanto detto a proposito del suono.

Anche se potrà sembrare ingenuo ai più esperti, compiti apparentemente semplici come questi appena descritti possono presentare problemi e trabocchetti per chi è completamente "digiuno" di Shell e del Dos in generale. Un consiglio, dunque, al nostro lettore ed agli altri come lui: approfondire la conoscenza del DOS spulciando ben bene il manuale, e magari rintracciando gli arretrati della nostra rivista dal n. **75** al n. **83**, nei quali i comandi AmigaDOS sono stati sviscerati in tutti i loro aspetti.

In arrivo... con dote

Sono un ragazzo di 16 anni, che sta per abbandonare il C/64 per Amiga. Vorrei sapere se potrò usare il mio attuale joystick anche su Amiga, e se potrò sostituirlo al mouse. Poi vorrei anche sapere se, comprando qualcosa come il "64 emulator" potrò adoperare la stessa stampante del C/64.
(Alberto Rusolo - Atripalda)

Per quanto riguarda il **joystick**, non dovrebbero sorgere problemi sul suo uso, ma non certo fino al punto di sostituire il mouse.

A meno che non sia specificamente costruito allo scopo di emularlo (come il **Mouse Stick**), nessun joy può implementare le prestazioni del "topo".

Quanto alla seconda domanda, un'altra di quelle ultra ricorrenti, non c'è che la solita, laconica risposta: a meno che la stampante non sia fornita di porta parallela **Centronics**, scordatelo!

...con **Chr$(69)** che potrà, come ovvio, essere sostituito da un qualunque codice (anche stringa, per esempio "**e**") adatto allo scopo desiderato.

Scanner e manuale

Ho acquistato lo scanner della Golden Image da voi recensito sul n. 83. Vorrei sapere se esiste una versione italiana del manuale di Touchup.
(Franco Landi - Firenze)

Al momento non ci risulta sia stato tradotto.

Se mai ciò avverrà, sarà nostra premura comunicarlo in queste pagine.

Monitor e sfarfallio

Il monitor 1084 Commodore è il migliore supportabile da Amiga 500, o posso ottenere risultati superiori, eliminando o almeno limitando lo sfarfallio delle immagini quando si lavora in alta risoluzione?

(Paolo Caccavo - Napoli)

Prestazioni grafiche superiori a quelle del monitor 1084 possono essere implementate senza problemi su Amiga 500, ma come ovvio a costi decisamente maggiori.

Per eliminare lo sfarfallio, tra l'altro, non è sufficiente cambiare il monitor.

L'ideale sarebbe dotare il computer di una scheda **Flicker Free**, cui accoppiare un monitor **Multisync**.

Se, come sembra il caso del lettore, si sfrutta Amiga per lavori in **Cad**, la soluzione è più che indicata.

Basic e Assembler

Ho letto sul manuale del Basic che si può adoperare l'istruzione Call per far eseguire dei sottoprogrammi in linguaggio macchina, inviando dei parametri racchiusi tra parentesi. Vorrei usare questo in un mio programma, ma non ho capito come deve fare il sottoprogramma per leggere quei parametri.
(Maurizio Carini - Milano)

L'argomento richiederebbe una trattazione molto più completa di quanto possibile

Speciale Principianti

Strane maiuscole

Ho da poco un Amiga, e ho cominciato a comprare qualche rivista (come la vostra) per imparare qualcosa. Una curiosità: perché vedo spesso scritte parole con una maiuscola in mezzo? L'ho notato diverse volte, e in diverse riviste, così ho pensato che non possono essere errori di stampa...
(Ciro Conti - Napoli)

In effetti non si tratta di errori di stampa, o almeno non sempre (...), quanto piuttosto di una convenzione molto adoperata su Amiga. Va detto che la cosa ha due aspetti: uno tecnico, ed uno più generale.

Il primo, riguarda per esempio i vari nomi di funzioni di libreria e cose similari, che proprio la documentazione ufficiale definisce adoperando delle maiuscole strane.

Un esempio: proprio in queste pagine si fa uso di una certa **AllocMem**, il cui significato qui non ha importanza. Perché quella *emme* maiuscola? Semplice: il termine, in effetti, è una contrazione che starebbe per qualcosa come Alloca Memoria (in realtà in inglese, ma il concetto non cambia).

Adoperare uno spazio, però, creerebbe problemi a compilatori e assemblatori che adoperano espressamente quella terminologia, e usare qualcosa come **Alloc_mem**, pur se teoricamente valido, in effetti adopererebbe un simbolo (la barretta in basso) che a sua volta per convenzione assume altri significati per il programmatore (e per il software specialistico).

Lo stesso problema, pur se con toni meno categorici, si può poi presentare in un ambito meno specifico.

Come ben sa chiunque abbia avuto modo di smanettare con AmigaDOS, se per esempio si assegna ad un file un nome tipo **mio programma**, ovvero contenente uno spazio nel suo contesto, ci si imbatterà inevitabilmente in una marea di difficoltà. Legate soprattutto all'uso dei doppi apici per specificarlo come parametro di molti comandi dos, per non parlare se poi va digitato assieme ad un eventuale path.

Difficoltà a parte, non va poi trascurata la proverbiale pigrizia dello smanettone, che si troverebbe a dover digitare in continuazione i doppi apici, lo spazio, e chi più ne ha più ne metta.

Una soluzione, stavolta consentita, potrebbe essere quella di adoperare un unico termine come **Mio_programma**, ma è ormai universalmente accettato, sulla scia dell'equivalente convenzione "tecnica", l'uso delle maiuscole per separare i vocaboli composti.

In considerazione, anche, della indifferenza di Amiga sulla grandezza (**case**) dei caratteri.

Nel nostro caso, quindi, il nome del file diventerebbe **MioProgramma**, con somma felicità dei polpastrelli, e relativa infelicità dei correttori di una rivista, spiazzati dalle "stramberie" di quella masnada di amighi...

in questa sede, per cui ci limiteremo a brevi cenni su come procedere, dando per scontata una certa dimestichezza con l'Assebler, ed una buona conoscenza di Amiga Basic.

Il manuale, in effetti, riporta che si può invocare l'esecuzione di una routine in linguaggio macchina (lm) usando una istruzione Call Variabile&(lista parametri).

Variabile& sarà da intendersi come una variabile a **32** bit (long) contenente l'indirizzo ove risiede la subroutine lm, che può anche essere contenuta in linee **Data** lette ed assegnate ad una variabile stringa tramite un ciclo **For ... Next**.

In questo caso basterà ottenere poi l'indirizzo della variabile stringa con **Sadd**, assegnarlo a una variabile long, e usare quest'ultima per l'istruzione Call.

Venendo al tema specifico della richiesta: si supponga di voler inviare alla routine Lm (per un qualunque motivo) un certo indirizzo, che può essere quello di una variabile, di una struttura di dati, o altro. Da basic, si dovrà impartire qualcosa come...

```
Call Routine&(x&)
```

Dove dovrà andare a pescare il valore contenuto da **x&** la subroutine lm?

La risposta è più semplice di quanto si creda: nello **Stack**.

Occorrerà però valutare che in cima ad esso, al momento del lancio della sezione lm, saranno presenti 4 byte (una long word) utili al sistema per organizzare il ritorno al Basic.

Immediatamente "sotto" (per i non addetti, lo stack è come una catasta ove ogni nuovo elemento si "appoggia" sul precedente), si troverà il parametro passato dal Basic.

In pratica, la routine lm potrebbe cominciare con qualcosa come...

```
Move.l 4(sp), a0
```

...per acquisire nel registro **A0** il parametro trasmesso dal basic.

Questo, come ovvio, se non si è intervenuto in altro modo sullo stack.

E' frequente, infatti, che le prime istruzioni di una routine provvedano a preservare il contenuto di alcuni (o tutti) i registri salvandolo nello stack, per poi ripristinarlo prima della conclusione. Per esempio, la routine potrebbe contenere come prima istruzione...

```
Movem.l a0/d0,-(sp)
```

In questo caso, occorrerà tenere presente che sullo stack sono stati aggiunti due valori long, ovvero 8 byte. Il parametro del basic, quindi, si troverà stavolta a 8 + 4 = **12** byte di distanza.

Per acquisirlo nel registro A0, sarà dunque necessario stavolta ricorrere a...

```
Move.l 12(sp),a0
```

... mentre la routine si concluderà con un normale ripristino dei registri, ovvero...

```
Move.l (sp)+,a0/d0
rts
```

Trattandosi di linguaggio macchina, inutile aggiungere che è indispensabile una certa attenzione se si vuole evitare l'intervento di Guru.

Quanto qui descritto, inoltre, non può certo esaurire un argomento più complesso di quanto si creda: appuntamento, dunque, in sedi più opportune, molto presto disponibili nelle pagine della rivista.

Ecco un "**indice**" degli argomenti (di recente pubblicazione) che riteniamo di maggior interesse per gli utenti della nostra rivista. La suddivisione per **argomenti** faciliterà la ricerca degli articoli; il **nome** di questi non sempre corrisponde a quello originale, per far meglio comprendere l'argomento trattato. Il numero in neretto corrisponde al **numero del fascicolo** della rivista.

SYSTEMS EDITORIALE PER TE

La voce
Aggiunge al C/64 nuovi comandi Basic che consentono sia di far parlare il computer, sia di farlo Cantare! Diversi esempi allegati.
Cassetta: L. 12000 - Disco: L. 15000

Raffaello
Un programma completo per disegnare, a colori, con il C/64: linee, cerchi, quadrati, eccetera. Valido sia per disegno a mano libera che geometrico.
Cassetta: L. 10000

Oroscopo
Devi solo digitare la data di nascita e le coordinate geografiche del luogo che ti ha dato i natali. Vengono quindi elaborate le varie informazioni (case, influenze dei segni astrali, eccetera) e visualizzato un profilo del tuo carattere. Valido per qualsiasi anno, è indicato sia agli esperti sia ai meno introdotti. E' allegata una tabella delle coordinate delle più note città italiane e l'elenco delle ore legali in Italia dal 1916 al 1978.
Cassetta: L. 12000 - Disco: L. 12000

Computer Music
Cassetta contenente numerosi brani di successo da far eseguire, in interrupt, al tuo C/64 sfruttando, fino in fondo, il suo generatore sonoro (SID).
Cassetta: L. 12000

Gestione Familiare
Il più noto ed economico programma per controllare le spese e i guadagni di una famiglia.
Cassetta: L. 10000 - Disco: L. 10000

Banca Dati
Il più noto ed economico programma per gestire dati di qualsiasi natura.
Cassetta: L. 10000 - Disco: L. 10000

Matematica finanziaria
Un programma completo per la soluzione dei più frequenti problemi del settore.
Cassetta: L. 10000 - Disco: L. 20000

Analisi di bilancio
Uno strumento efficace per determinare con precisione i calcoli necessari ad un corretto bilancio.
Cassetta: L. 10000 - Disco: L. 20000

Corso di Basic
Confezione contenente quattro cassette per imparare velocemente le caratteristiche delle istruzioni Basic del C/64 e i rudimenti di programmazione. Interattivo.
Cassetta: L. 19000

Corso di Assembler
Un corso completo su cassetta per chi ha deciso di abbandonare il Basic del C/64 per addentrarsi nello studio delle potenzialità del microprocessore 6502. Interattivo.
Cassetta: L. 10000

Logo Systems
Il linguaggio più facile ed intuitivo esistente nel campo dell'informatica; ideale per far avvicinare i bambini al calcolatore.
Diversi esempi allegati.
Cassetta: L. 6500

Compilatore Grafico Matematico
Uno straordinario programma compilatore, di uso semplicissimo, che permette di tracciare, sul C/64, grafici matematici Hi-Res ad altissima velocità. Esempi d'uso allegati.
Cassetta: L. 8000

Emulatore Ms-Dos e Gw-Basic
Un prodotto, unico nel suo genere, che permette di usare, sul C/64 dotato di drive, la sintassi tipica del più diffuso sistema operativo del mondo. Ideale per studenti.
Solo su disco: L. 20000

Emulatore Turbo Pascal 64
Permette di usare le più importanti forme sintattiche del linguaggio Turbo Pascal (anche grafiche!) usando un semplice C/64 dotato di drive.
Ideale per studenti.
Disco: L. 19000

Speciale drive
Questo speciale fascicolo costituisce una guida di riferimento per le unità a disco del C64/128.
Comprende anche un velocissimo turbo-disk più la mappa completa della memoria del drive.
Fascicolo + disco: L. 12000

Utility 1
Un dischetto pieno zeppo di programmi speciali per chi opera frequentemente con il drive.
Disco: L. 12000

Utility 2
Seconda raccolta di utility indispensabili per realizzare sofisticate procedure di programmazione.
Disco: L. 15000

Graphic Expander 128
Per usare il C/128 (in modo 128 e su 80 colonne) in modo grafico Hi-res. Aggiunge nuove, potenti istruzioni Basic per disegnare in Hi-Res con la massima velocità in modalità 80 colonne.
Disco: L. 27000

Directory
Come è noto, a partire dal N. 10 di "Software Club" (la rivista su disco per l'utente dei "piccoli" computer Commodore), vengono riportati tutti i listati, in formato C/64-C/128, pubblicati su "Commodore Computer Club".
In precedenza tali listati venivano inseriti, mensilmente, in un dischetto, di nome "Directory", che oltre ai programmi di C.C.C. ospitava decine di altri file tra cui musiche nell'interrupt, giochi, listati inviati dai lettori e altro.
Ogni disco, dal prezzo irrisorio, contiene quindi una vera miniera di software. Ordinando i dischetti di "Directory" si tenga conto che al N. 1 corrispondeva il contenuto del N. 34 di "Commodore Computer Club", al N. 2 il N. 35 e così via.
Ogni dischetto: L. 10000

Super Tot '64
La nuova e completa edizione del programma Tot 13 con tutti i sistemi di riduzione e di condizionamento.
Ampia sezione dedicata alla teoria.
fascicolo + disco: L. 15000

Amiga Totospeed
Finalmente anche per Amiga un programma orientato alla compilazione delle schedine totocalcio.
Fai tredici conil tuo Amiga.
disco: L. 20000

SYSTEMS EDITORIALE PER TE

Disk'o'teca

Grazie a questa nutrita raccolta di brani musicali potrete divertirvi ascoltando i migliori brani prodotti dai vostri beniamini, oltre a una serie di composizioni prodotte "in casa".
In omaggio un bellissimo poster di Sting.

Disco: L.15.000

Assaggio di primavera

Esclusivo!
In un'unica confezione potrete trovare ben due cassette di videogiochi assieme a un comodo e funzionale joystick.

Cassette: L. 15.000

LIBRI TASCABILI

64 programmi per il C/64

Raccolta di programmi (giochi e utilità) semplici da digitare e da usare. Ideale per i principianti. **(126 pag.)**

L. 4800

I miei amici C/16 e Plus/4

Il volumetto, di facile apprendimento, rappresenta un vero e proprio mini-corso di Basic per i due computer Commodore. Numerosi programmi, di immediata digitazione, completano la parte teorica. **(127 pag.)**

L. 7000

62 programmi per C/16, Plus/4

Raccolta di numerosi programmi, molto brevi e semplici da digitare, per conoscere più a fondo il proprio elaboratore.
Ideale per i principianti. **(127 pag.)**

L. 6500

Micro Pascal 64

Descrizione accurata della sintassi usata dal linguaggio Pascal "classico". Completa il volume un programma di emulazione del PL/O sia in formato Microsoft sia in versione C/64 (da chiedere, a parte, su disco). **(125 pag.)**

L. 7000

Dal registratore al Drive

Esame accurato delle istruzioni relative al'e due più popolari periferiche del C/64. Diversi programmi applicativi ed esempi d'uso. (94 pag.)

L. 7000

Il linguaggio Pascal

Esame approfondito della sintassi usata nel famoso compilatore. (112 pag.)

L. 5000

Simulazioni e test per la didattica

Raccolta di numerosi programmi che approfondiscono e tendono a completare la trattazione già affrontata sul precedente volume. **(127 pag.)**

L. 7000

Dizionario dell'Informatica

Dizionario inglese-italiano di tutti i termini usati nell'informatica. (Edizione completa). **(385 pag.)**

L. 10000

Word processing: istruzioni per l'uso

Raccolta delle principali istruzioni dei più diffusi programmi di w/p per i sistemi Ms-Dos: Word-Star, Samna, Multimate Advantage, Word 3. **(79 pag.)**

L. 5000

Unix

Un volumetto per saperne di più sul sistema operativo professionale per eccellenza.
Un necessario compendio per l'utente sia avanzato che inesperto **(91 pag.)**

L. 5000

ABBONAMENTO

Computer Club
11 fascicoli: L. 60.000

ARRETRATI

Ciascun numero arretrato di C.C. L. 6.000

Come richiedere i prodotti Systems

Coloro che desiderano procurarsi i prodotti della Systems Editoriale devono inviare, oltre alla cifra risultante dalla somma dei singoli prodotti, L. 3500 per spese di imballo e spedizione, oppure L. 6000 se si desidera la spedizione per mezzo raccomandata.
Le spese di imballo e spedizione sono a carico della Systems se ciascun ordine è pari ad almeno L. 50000.
Per gli ordini, compilare un normale modulo di C/C postale indirizzato a:

C/C Postale N. 37 95 22 07
Systems Editoriale Srl
Via Mosè, 22
20090 Opera (MI)

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento"), non solo il vostro nominativo completo di recapito telefonico, ma anche i prodotti desiderati ed il tipo di spedizione da effettuare.
Per sveltire la procedura di spedizione sarebbe opportuno inviare, a parte, una lettera riassuntiva dell'ordine effettuato, allegando una fotocopia della ricevuta del versamento.
Chi volesse ricevere più celermente la confezione deve inviare la somma richiesta mediante assegno circolare, oppure normale assegno bancario (non trasferibile o barrato due volte) intestato a:

Systems Editoriale
Milano

AMSTRAD PC 3386SX

IL "386" CHE FA NOTIZIA

"Uno degli SX più veloci che abbiamo visto".

PC WORLD MAGAZINE

"Questa è una macchina sicura e ben progettata".

WHAT PC MAGAZINE

"Questi prezzi daranno filo da torcere da IBM in giù".

WHAT MAGAZINE

"Un protagonista scintillante, notevolmente più veloce degli altri".

PC PLUS MAGAZINE

PC3386SX HD 12 MD 1 MB RAM
HD 40 MB MONITOR VGA

L. 2.790.000 + IVA

Scopri i computer Amstrad della terza generazione. Flessibilità e potenza, velocità ed espandibilità sono alcune delle qualità migliori della nuovissima gamma Amstrad. Puoi far convivere drive di formato diverso (3" e/o 5") senza occupare slot di espansione, dimensionare la RAM secondo le tue necessità, soddisfare le tue esigenze più particolari, grazie a 5 slot disponibili.

Li trovi qui. Presso tutti i rivenditori Leader Amstrad (li trovi su Amstrad Magazine in edicola), oppure telefona a Pronto Amstrad 02/3263210, avrai tutte le informazioni che desideri.

Cognome ____________________
Nome ____________________
Via ____________________ n. ______
Cap ______ Città ____________________

Cod. 3386. Tagliate lungo la linea tratteggiata e spedite a: Amstrad via Riccione 14 - 20156 Milano.

Modello	CPU	RAM	Drives	Grafica	Slots	Prezzo IVA esclusa
PC 3086 SD 12MD	8086 8 MHz	640Kb	1 FD 720Kb	P-VGA	4	1.190.000
PC 3086 HD 12MD	8086 8 MHz	640Kb	1 FD 720 Kb HD 30 Mb	P-VGA	4	1.790.000
PC 3286 SD 12MD	80286 16 MHz	1 Mb	1 FD 1.44 Mb	P-VGA	5	1.790.000
PC 3286 HD 12MD	80286 16 MHz	1 Mb	1 FD 1.44 Mb HD 40 Mb	P-VGA	5	2.290.000
PC 3386SX HD12MD	80386SX 20MHz	1 Mb	1 FD 1.44 Mb HD 40 Mb	P-VGA	5	2.790.000

AMSTRAD GENERAZIONE 3; estratto listino prezzi.

OPINIONE Comunicazione Integrata